3D-XPath: High-Density Managed DRAM Architecture with Cost-effective Alternative Paths for Memory Transactions

Sukhan Lee
Seoul National University
Samsung Electronics
sh1026.lee@scale.snu.ac.kr

Kiwon Lee
Samsung Electronics
kiwon77.lee@samsung.com

Minchul Sung
Seoul National University
minchul.sung@scale.snu.ac.kr

Mohammad Alian
UIUC
malian2@illinois.edu

Chankyung Kim
Samsung Electronics
chank.kim@samsung.com

Wooyeong Cho
Samsung Electronics
solarmac.cho@samsung.com

Reum Oh
Samsung Electronics
reum.oh@samsung.com

Seongil O
Samsung Electronics
seongil.o@samsung.com

Jung Ho Ahn
Seoul National University
gajh@snu.ac.kr

Nam Sung Kim
UIUC
Samsung Electronics
nskim@illinois.edu

ABSTRACT

The advance of DRAM manufacturing technology slows down, whereas the density and performance needs of DRAM continue to increase. This desire has motivated the industry to explore emerging Non-Volatile Memory (e.g., 3D XPoint) and the high-density DRAM (e.g., Managed DRAM Solution). Since such memory technologies increase the density at the cost of longer latency, lower bandwidth, or both, it is essential to use them with fast memory (e.g., conventional DRAM) to which hot pages are transferred at runtime. Nonetheless, we observe that page transfers to fast memory often block memory channels from servicing memory requests from applications for a long period. This in turn significantly increases the high-percentile response time of *latency-sensitive* applications. In this paper, we propose a high-density managed DRAM architecture, dubbed 3D-XPath for applications demanding both low latency and high capacity for memory. 3D-XPath DRAM stacks conventional DRAM dies with high-density DRAM dies explored in this paper and connects these DRAM dies with 3D-XPath. Especially, 3D-XPath allows unused memory channels to service memory requests from applications when primary channels supposed to handle the memory requests are blocked by page transfers at given moments, considerably increasing the high-percentile response time. This can also improve the throughput of applications frequently copying memory blocks between kernel and user memory spaces. Our evaluation shows that 3D-XPath DRAM decreases high-percentile

response time of latency-sensitive applications by ~30% while improving the throughput of an I/O-intensive applications by ~39%, compared with DRAM without 3D-XPath.

CCS CONCEPTS

• **Hardware** → **Dynamic memory**; *Die and wafer stacking*; *Memory and dense storage*;

KEYWORDS

3D stacked memory, asymmetric latency memory, hardware managed migration

1 INTRODUCTION

The demand for larger main memory capacity keeps increasing. However, as DRAM technology scaling slows down, so does the capacity scaling of a Dual-Inline Memory Module (DIMM) consisting of DRAM packages. This in turn constrains the capacity of main memory systems, because the maximum number of DIMMs per memory channel (or simply “channel”) is limited to a small number, and it decreases with higher signaling rates. In such a case, it becomes hard to further increase the capacity without sacrificing either bandwidth or latency. Although *buffered* DIMMs such as Registered DIMMs [29] and Load-Reduced DIMMs [28] allow more DIMMs per channel without sacrificing the bandwidth, they increase the latency.

To improve capacity per DRAM package, we may consider stacking DRAM dies. However, as the number of stacked dies increases, (1) the cost increases superlinearly especially due to drop in yield [23] and (2) the data transfer rate per TSV, which connects these dies, decreases due to worsened skews in setup/hold timings [49]. This challenge has motivated the industry to explore emerging Non-Volatile Memory (NVM) technology such as 3D XPoint or high-density DRAM such as Managed DRAM Solution (MDS) [1]. Nonetheless, both NVM and high-density DRAM increase their capacity at the cost of longer latency, lower bandwidth, or both. Therefore, it is essential to use such slow memory with fast memory (such as

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

PACT '18, November 1–4, 2018, Limassol, Cyprus

© 2018 Association for Computing Machinery.

ACM ISBN 978-1-4503-5986-3/18/11...\$15.00

<https://doi.org/10.1145/3243176.3243191>

Figure 1: Distributions of Apache response time when main memory systems are built with heterogeneous DRAM where (1) hot-page transfers to faster DRAM block memory requests (C1), and (2) where hot-page transfers do not block memory requests (C2).

conventional DRAM) and transfer hot pages to the fast memory to minimize the negative impact of prolonged latency and/or low bandwidth of the slow memory on performance [56, 61].

Analyzing On-Line Data-Intensive (OLDI) applications which often require high capacity and low latency memory [10, 34], however, we observe that page transfers (or swaps) between fast and slow memories frequently block channels from servicing memory requests of the applications for a long period. This blocking in turn unacceptably increases the high-percentile response time of latency-sensitive OLDI applications. To quantify this effect, we evaluate two configurations stacking slow DRAM with fast DRAM: (C1) one may block memory requests from the Apache web server [3] and (C2) the other does not during page transfers (see Figure 1; Section 5 details the experimental setups). The 95th-percentile and average response time values of C2 are 32% and 24% lower than those of C1, respectively.

Tackling this challenge, we propose a high-density managed DRAM architecture, 3D-XPath, for applications that require both high capacity and low latency for memory. Specifically, advocating the use of high-density DRAM with conventional DRAM, we first discuss challenges in DRAM technology scaling and then explore design space of high-density DRAM. Second, we propose 3D-XPath, which provides alternative paths for the following two purposes: (P1) efficient transfers of pages between dies; and (P2) prompt services of memory requests from applications (“memory requests” hereafter) when primary channels for the memory requests are already transferring pages. To cost-effectively offer 3D-XPath, we exploit the following two observations from our experiment and in-depth analysis of industry 3D-stacked DRAM [35]. (O1) One or more channels are often unused for a certain period. (O2) A set of I/O TSVs constituting each channel is *physically* connected to the I/O connection points of all stacked dies, and it can be *electrically* connected to any stacked die at a time by controlling tri-state buffers with decoder logic. In a 3D-stacked memory package with multiple channels, O1 and O2 allow a memory controller to dynamically establish an alternative channel for a memory request to one rank although its primary channel, which is shared with other ranks, is used by a page transfer to other ranks.

In this paper, we evaluate 3D-XPath for memory which stacks conventional and high-density DRAM dies and uses the conventional DRAM as hardware-managed cache. Nonetheless, 3D-XPath is also applicable to memory which (1) stacks any heterogeneous

memory dies including DRAM and NVM and/or (2) uses conventional DRAM as a hardware-managed cache with large cache lines. Lastly, 3D-XPath can greatly improve performance of I/O-intensive applications that frequently need to copy memory blocks between kernel and user memory spaces, as 3D-XPath can efficiently support in-memory copy operations.

To evaluate the effectiveness of 3D-XPath, we model a heterogeneous 3D-stacked memory system with two conventional and six high-density DRAM dies. As our baseline system, we use a 16-core chip-multiprocessor system with the heterogeneous 3D-stacked memory system. Through system-level simulations we compare heterogeneous memory systems adopting our proposed techniques with the baseline system. Our evaluation shows that heterogeneous 3D-stacked DRAM with 3D-XPath reduces 95th-percentile response time of OLDI applications by ~30% while improving throughput of an I/O-intensive application by ~39%. Lastly, heterogeneous 3D-stacked DRAM can provide higher capacity than homogeneous 3D-stacked DRAM with only conventional DRAM for the same number of dies, but it still gives worse 95th-percentile response time even for applications demanding high memory capacity unless it adopts 3D-XPath, according to our evaluation.

2 BACKGROUND

2.1 Heterogeneous Main Memory Systems

In a heterogeneous main memory system, DRAM can be deployed as hardware- or software-managed cache, both posing their own challenges. DRAM as hardware-managed cache (e.g., [40, 43]) needs memory space to store tags and decide where to place the tags. Furthermore, DRAM used as hardware-managed cache could be slower than DRAM used as main memory (e.g., cache mode in Intel Knights Landing (KNL) [63]), due to the latency penalty of comparing tags for every memory request. Lastly, since DRAM used as hardware-managed cache is not a part of main memory address space, it is less desirable for memory capacity sensitive workloads than DRAM used as main memory.

Alternatively, a heterogeneous memory can constitute unified memory space (e.g., a flat mode in KNL [63]), and let the OS or applications explicitly transfer hot pages to DRAM. However, it is still an active research topic to identify hot pages and when to transfer them to DRAM [53, 54, 67]. Lastly, software-managed cache approaches for heterogeneous memory systems often need to transfer large memory blocks to DRAM to handle transfer pages. This additional challenge prevents channels from servicing memory requests for a prolonged period and thus significantly increases the high-percentile response time of latency-sensitive applications.

2.2 Specialized DRAM

Due to many manufacturing challenges, it is becoming harder to improve latency, bandwidth, and capacity of DRAM altogether [42]. However, by not trying to improve every primary aspect of DRAM metrics altogether, it gets easier to make DRAM specialized in certain aspects, such as latency or capacity. As an example, we may consider low-power DRAM as LPDDRx for mobile domain and GDDRx for bandwidth-demanding graphic domain. The latency of DRAM is primarily determined by the time (T1) to deliver command and data signals through global interconnects which traverse

Figure 2: A standard 3D-stacked DRAM 2.5D-integrated with CPU. DRAM dies are connected to a logic die through TSVs and micro bumps; a logic die communicates with memory controllers through an interposer.

DRAM banks; (T2) to sense voltage developed by charge sharing of a DRAM cell and its corresponding bitline (local datapath); and (T3) to precharge the bitlines (BLs) if needed. Son *et al.* [65] analyzed that the perimeter of a DRAM die determines T1 whereas the BL capacitance determines T2 and T3. As we populate more BL sense amplifiers (BSAs), we can reduce the latency (e.g., Reduced-Latency DRAM [45]) at the cost of sacrificing the capacity.

2.3 3D-stacked Memory

Recently, the industry starts to adopt various 3D-stacked DRAM, such as DDR4 3DS [18], Hybrid Memory Cube (HMC) [52], High Bandwidth Memory (HBM) [35], and Multi-Channel DRAM (MC-DRAM) [63] for servers and throughput computing. In this work, we take MCDRAM or HBM as a standard 3D-stacked DRAM (Figure 2). Processor units are connected to 3D-stacked DRAM dies through a silicon interposer. A stack consists of several DRAM dies (typically 4 or 8, by default 8 in this paper) and a logic die. Each datapath TSV (and silicon interposer interconnect) transfers data at 2 Gbps, the transfer rate of HBM2 standard [27]. 512 datapath TSVs per stack lead to 128 GB/s of bandwidth. A stack has multiple channels (4 by default) to utilize this sheer bandwidth efficiently. A DRAM die is connected to one or few channels (2 by default).

We assume that all DRAM dies are fabricated identically to optimize the manufacturing cost like HBM. Then, I/O TSVs constituting four channels are *physically* connected to the I/O connection points of all stacked DRAM dies. A set of I/O TSVs constituting a channel can be *electrically* connected to one of the stacked dies by controlling tri-state buffers with decoder logic (Figure 2(left)). The decoder logic can enable or disable the tri-state buffers, and the manufacturer uses *e*-fuses to program a Stack ID (SID) into the decoder logic of a stacked DRAM die in the standard 3D-stacked memory. This electrically connects the I/O connection points of a stacked DRAM die to a specific set of I/O TSVs establishing a channel at a manufacturing step. Furthermore, examining the I/O pin layout of the 3D-stacked DRAM, we see that I/O TSVs of all channels associated with the same bit index are closely placed (Figure 2(right)).

TSVs are placed at the center of dies to minimize the worst case topological distance between DRAM cells and TSVs and thus latency [35] [27]. A DRAM die consists of multiple bank groups (8 bank groups per die, 4 bank groups per channel, by default) and

each group consists of multiple banks (4 by default). A DRAM bank is organized and operates conventionally; a row to access is first activated at BSAs (*i.e.*, a row buffer) and then column addresses accompany read/write commands to access data in the row. Within a DRAM die, each bank operates independently except that 1) the number of bank activates within a certain interval is limited due to timing constraints such as t_{FAW} to restrict a surge in current draws which lead to fluctuation in voltage levels, and 2) all banks mapped to a channel cannot transfer data concurrently due to structural hazard. The bank is structured hierarchically; each bank group has a separate inter-bank dataline to facilitate higher data transfer rate for the transactions heading to different bank groups. A channel and the corresponding bank groups are connected through a multiplexer (mux). The logic die repeats address, command, and data signals between off-stack controllers and DRAM dies.

3 HIGH-DENSITY DRAM ARCHITECTURE

In contrast to low-latency DRAM proposals [45, 60, 65], high-density DRAM, another important class of specialized DRAM designs, is relatively less explored. As latency and bandwidth gap between DRAM and NAND flash increases, the industry introduces solutions to fill this gap, dubbed Storage Class Memory (SCM) [62]. Examples include single-level-cell NAND which focuses on reducing latency (e.g., SanDisk ULLtra DIMMS [14]) and Intel Optane which exploits 3D XPoint technology. A less elaborated but intriguing solution is to specialize DRAM designs for higher density [42]. We identify the key design challenges of high-density DRAM and sketch its plausible design solutions.

3.1 Key Design Challenges

DRAM technology scaling slows down primarily because the DRAM industry is demanded to improve every major aspect of DRAM metrics: latency, bandwidth, and capacity [41, 42]. That is, the DRAM industry can still scale the dimensions of transistors, capacitors, and interconnects (*i.e.*, improving density) for a few more generations like logic and Flash technologies, but it cannot do so while maintaining or improving latency and bandwidth.

A DRAM cell consists of a capacitor and an access transistor. A DRAM subarray is a 2D array of DRAM cells. DRAM stores data

Figure 3: The impact of DRAM technology scaling on process parameters, timing, and density.

by charges in capacitors, which are detected by sense amplifiers after sharing their charges with vertical bitlines (BLs). A horizontal wordline (WL) controls the access transistors of a DRAM row. Because the number of cells connected to a WL surpasses several thousands in modern DRAM chips, a WL takes a hierarchical structure, consisting of several local WLs. A mat is a unit of DRAM where cells are connected through local WLs and BLs.

A large surface area is needed to increase the capacitance of a DRAM cell, making non-planar designs less viable than NAND Flash where 3D designs are gaining popularity rapidly. Therefore, primary ways to improve storage density of this planar DRAM design is either to increase cell efficiency or to keep scaling down fabrication technology. Increasing cell efficiency (a portion of aggregated cell area over an entire DRAM chip area) has limited potential because a DRAM chip has inevitable circuitry such as inter-bank datalines, off-chip I/O buffers, electrostatic discharge protection, and charge pumps. Moreover, robust sensing requires a certain degree of voltage difference after charge sharing (delta V), limiting the number of cells shared through a BL.

Beyond 20nm process nodes denoted by '1Xnm',¹ '1Ynm,' and '1Znm' [15, 42], however, DRAM technology scaling faces three major process challenges. First, fine-pitch metal line and contact resistance (R) increase drastically. The metal resistivity sharply increases below 20nm film thickness because of shrinking in metal volume and the surface scattering effect [51]. This increases the resistance of WLs and BLs. Second, the capacitance of a DRAM cell decreases because cells become smaller and it gets harder to further increase the aspect ratio (or height) of a cell capacitor [42]. Third, the ratio of faulty cells increases as DRAM cell area becomes smaller and process variation is exacerbated [15].

Significant efforts have made on various aspects of DRAM designs including material (e.g., high-k metal gate to increase transistor speed), fabrication technology (e.g., filling air within spacers to reduce BL capacitance [51]), and cell structuring (e.g., deploying cells like honeycomb to increase cell spacing [51]) to increase DRAM density *without sacrificing timing constraints*. However, those incur substantial manufacturing costs, need time to be applied stably, or become one-time *magic* desiring a new solution (another *magic*) next time. Therefore, without relying on material or process breakthrough, an alternative solution for higher density is to continue DRAM technology scaling while relaxing key timing constraints, which are affected by the aforementioned challenges.

Figure 4: The impact of different resistance and capacitance on timing parameters. When the WL resistance is doubled and tripled, the WL signal is delayed by 2 ns and 4 ns, respectively. When the BL resistance is doubled and its capacitance is decreased by 30%, the BL signal is delayed by 1 ns.

3.2 Plausible High-density DRAM Designs

These process challenges can be overcome with the help of relaxing tight timing constraints and architectural support. Increase in resistance of fine-pitch metal lines and contacts slows signal transfer speed and causes timing failure. For example, higher WL resistance delays the delivery of an activation signal to cell transistors located at the far end of a DRAM subarray. That cell transistor turns on slowly; by relaxing tRCD (the minimal interval from an activate to a read/write command on a DRAM bank), a sufficient amount of charge sharing time can be secured. Similarly, the effects of high BL and contact resistance can be suppressed by relaxing key DRAM timing parameters, such as tRP (BL precharge time), tWR (time to write data to DRAM cells), and tRAS (time to destructively read data from a DRAM cell and then restore the data back), because this datapath resistance within a subarray directly affects data restore time and BL precharge delay.²

Figure 3 projects the impact of primarily scaling the dimensions of transistors, capacitors, and interconnects on major DRAM design parameters as well as timing parameters and storage density. Here, we expect no significant innovation (*magic*) in DRAM process technology based on our evaluation setup with the following modeling details. The cell mat, WL drivers, and sense amplifiers critically affect DRAM timing parameters. The state-of-the-art 6F² DRAM cell structure has one transistor and one capacitor per crossing point of a WL and a BL. We modeled the critical path by choosing the farthest cell from WL drivers and sense amplifiers of a DRAM mat and conducted SPICE simulation. The voltage-timing diagrams on Figure 4(a) and (b) show SPICE simulation results corresponding to the change of WL resistance, when a WL is developed from Vss (often ground) to Vpp (WL activation voltage) by the WL driver (activation) and from Vpp to Vss (precharge). When the resistance of the WL is doubled and tripled, the signal is delayed by 2 ns and 4 ns, respectively. Figure 4(c) shows simulation result of BL voltage changing from Vss to Vcore (high voltage levels in DRAM cells) according to the change of BL resistance. When the BL resistance is doubled and the BL capacitance is decreased by 30%, the signal is delayed by 1 ns. This shows that the DRAM industry may double the density over two generations (1Xnm to 1Znm) at the cost of increasing key timing parameters such as tRCD, tRAS, and tRP by

¹Notations such as 1X and 1Y are widely used terms in DRAM vendors and X, Y, and Z (X > Y > Z) are single-digit numbers, where specific values depend on vendors.

²A surge in resistance of fine-pitch metal lines and contacts does not affect tCL (read command to first data delay) much as global control and datapath use thicker (non-minimal pitch) wires having much lower resistance compared to local WLs and BLs.

28.6% (4 ns due to WL turn-on signal propagation delay), 13.3% (6 ns by BL develop delay and tRCD increment), and 42.8% (WL turn-off signal propagation delay and BL precharge delay).

Reduction in cell capacitance (second challenge) decreases data retention time. More leaky cells have retention time below DRAM refresh window (tREFW). Increase in these leaky and faulty (third challenge) DRAM cells necessitates stronger redundancy. This problem can be greatly alleviated by recently proposed or applied reliability schemes, such as In-DRAM ECC [15], ArchShield [47], CiDRA [64], XED [48], and Bamboo ECC [36]. These provide cost-effective solutions to common single-bit failures compared to traditional solutions, such as provisioning spare DRAM rows and columns. These schemes typically increase DRAM access latency. For example, In-DRAM ECC, which we assume in this paper, requires error checking before transmitting data out of DRAM banks, increasing tCL (Table 1).

The aforementioned techniques enable us to get the capacity merit over DRAM taking the conventional evolution path at the cost of lower performance. The pace in reduction of DRAM cell capacitance is faster than that of BL capacitance; and hence more BLSAs should be populated at a given DRAM capacity, increasing area overhead. However, by relaxing timing constraints, BL capacitance can further be reduced; if we make BLs narrower and increase the spacing between BLs, a BL has higher resistance (e.g., deteriorating tWR) but lower capacitance. This helps the voltage difference after charge sharing (delta V) mostly unchanged even after technology scaling (Figure 3) and restricts the rise of area overhead due to populating more BLSAs.

Strong reliability solutions demand additional (parity) DRAM cells (e.g., 6.25% in [15]). Still, storage density improvement (in absolute area) outweighs the costs induced by these techniques. Assuming that these techniques advance the DRAM technology node by two generations compared to the conventional DRAM dies, high-density DRAM can halve die area for a given capacity as the half-pitch size is reduced by 20% on average per DRAM technology shrink. Either the number of rows, columns, or banks should be increased to deal with more capacity provided by technology scaling. In this paper, we assume that the number of banks within a bank group is doubled. This will simplify support for page swapping, which will be further elaborated in Section 4.2. In summary, we can increase the density by 2.3 $\times$ at the cost of increasing tRCD, tRAS, and tRP by 29%, 13.3%, 43%, respectively. *That is, although we consider the cost of the reliability measures, we can double the capacity per DRAM die.*

4 3D-STACKED DRAM WITH ALTERNATIVE PATHS FOR MEMORY TRANSACTIONS

To build 3D-XPath DRAM, we propose to stack 8 DRAM dies like the standard 3D-stacked DRAM depicted in Section 2.3, but we replace top 6 DRAM dies with high-density DRAM dies (or Capacity-Optimized DRAM (CapD) dies hereafter) proposed in Section 3. This gives roughly 28 GB capacity or 1.75 $\times$ higher density than the standard 3D-stacked DRAM. Regardless of DRAM types, each DRAM die is connected to two channels and has the same number of bank groups (i.e., four) per channel. A conventional DRAM die (or Performance-Optimized DRAM (PerfD) die hereafter) has 4 banks

per bank group whereas a CapD die has 8 banks per bank group. The data bus width of a PerfD die (256 bits) is twice as wide as that of a CapD die. The size of a DRAM row/page is 4 KB for both PerfD and CapD types. The remainder of this section describes how we connect these DRAM dies with 3D-XPath and how we efficiently manage page transfers between CapD and PerfD while servicing memory requests with 3D-XPath.

4.1 3D-XPath Architecture

3D-XPath aims to efficiently service ordinary memory requests from processors while transferring pages between PerfD and CapD, for which we propose 3D-Path and XPath (together 3D-XPath).

3D-Path: A page swap or copy between PerfD and CapD requires many 64-byte (i.e., cache line) memory transactions, blocking memory requests to another bank group of CapD or PerfD involved with swapping/copying pages for a long period. This, in turn, significantly increases the latency of servicing the memory requests, hurting not only the overall system performance but also the high-percentile response time of latency-sensitive applications. Tackling this challenge, we leverage two key observations we made from our experiment and in-depth analysis of industry 3D-stacked DRAM. First, we observed that one or more channels are often unused for a certain period. To further support this observation, we monitor all memory transactions and measure the number of cycles for which a given channel stays unoccupied for the same simulation. At least one channel is empty during 8 cycles for more than 80% of total simulation time. This means that we can efficiently utilize these idle channels by 3D-Path. Second, all channels are physically connected to all DRAM dies, and a set of I/O TSVs constituting a channel can be *electrically* connected to any stacked die by controlling tri-state buffers with decoder logic (Section 2.3).

From these observations, we propose 3D-Path that diverts memory requests to a less frequently utilized channel at a certain time period, which is feasible with a simple adaptation of the standard 3D-stacked DRAM. Specifically, we can replace the multiplexers, which connect inter-bank datalines (one per bank group) to the channel I/O, with two crossbars (4:1 and 1:4 muxes and demuxes that connect bank groups to channels). Figure 5 shows the detailed implementation to enable 3D-Path. Each DRAM die has two enable signals per channel for normal memory transaction and migration. For example, a normal access is performed through the predefined TSV channel with en_0 to access bank group 0. A migration should be suspended until the current channel becomes idle without 3D-Path, as migration cannot use channel 0 which is already used for the normal access. With 3D-Path, the 3D-Path controller on the logic die controls the extra enable signal (en_1) on each die to send migration data on an idle channel (channel 3 in Figure 5). This allows non-blocking migration operation through the idle channel 3 with en_1 without interrupting the normal memory transaction on channel 1 with en_0. To find an idle channel, we implement a detection logic on the logic die; the logic detects how many cycles will be empty for each channel by a DRAM access command, and the tri-state buffer decoder on the logic die controls the enable signals to use 3D-Path. The crossbars incur little space and timing costs because the I/O TSVs of all channels associated with the same bit index are closely placed (Section 2.3). Our estimation using

Figure 5: 3D-stacked heterogeneous DRAM architecture, which consists of 2 PerfD and 6 CapD dies. PerfD dies have more TSVs for doubled datapath. Crossbar switches to connect TSV channels for 3D-XPPath locate at both CapD and PerfD. A logic die swaps hot pages between CapD and PerfD. The blue colored blocks are required to implement alternative paths.

CACTI-3DD and 20 nm DRAM technology information [57] shows that the cost of 3D-Path is 1.5% of a PerfD die. Also, the area cost of the enhanced tri-state buffer controller is less than 0.1% of the logic die.

XPPath: It is desirable to increase the transfer rate between a memory controller and PerfD, because effective page swaps lead to more frequent memory requests to PerfD and thus latency surges due to the queuing delay. There are two implementation options to increase the transfer rate. The first option is to increase the bit rate (operating frequency) of each pin/TSV. This can keep the number of TSVs unchanged, but it is difficult to implement due to worsened skews in setup/hold timings [49] as the bit rate increases. When the number of stacked dies increases, timing skews among datapath interconnects increase further and setup/hold timing margins shrink, limiting the operating frequency of TSVs (Section 2.3). The second option is to increase the number of datapath interconnects and TSVs. Specifically, we propose to double the datapath width only for PerfD because CapD is optimized for capacity and populating more TSVs hurts the density of CapD (Figure 6). As PerfD dies are located closer to a logic die than CapD dies, it is more tractable to handle timing skews among datapath interconnects [49].

Due to wider datapath (*i.e.*, additional DQ[255:128] per channel), the transfers to/from PerfD dies require shorter Burst Length (BL) than those to/from CapD dies. To match the transfer rate between CapD and PerfD, the logic die requires a 512-bit prefetch buffer per channel. This needs more TSVs and thus requires more die space. From prior work [35], we estimate that it requires 2% more die space. To keep up with the doubled bandwidth of PerfD, we also propose to increase the bit rate per pin between the logic die and the host memory controllers from 2 Gbps to 4 Gbps. To support the feasibility of increasing the bit rate through an interposer, we turn to existing I/O implementations: GDDR5X drives 10 to 14 Gbps per pin [30], and LPDDR4X is faster than 4 Gbps per pin on a noisier PCB channel environment than interposer [31]. We also conducted HSPICE signal integrity simulation with s-parameter and noise models that were specified in prior work [9], where we observe an enough eye diagram on interposer channels at 4 Gbps.

Figure 6: Structure and mechanism for data transfer between 3D-stacked DRAM components with doubled datapath. CapD and PerfD transfer data with 2 Gbps per dataline to the logic die. The logic die can transfer data to a memory controller at 4 Gbps when a PerfD is accessed because a PerfD has twice wider datapath than that between the logic die and the memory controller.

To further reduce the latency of page swaps, we exploit the doubled datapath and propose XPPath, an enhanced page-swap mechanism using wider datapath with two swap buffers per channel on the logic die (Figure 7). Two swap buffers receive pages from a source row (cold page) and a target row (hot page), respectively. CapD uses the first half of the doubled datapath (*i.e.*, DQ[127:0]) while PerfD uses the second half of 128 to 255 bit (*i.e.*, DQ[255:128]) to send two pages to the swap buffers. This transaction requires 128 tCK slots for a 4 KB DRAM page without interrupt. Subsequently, we simultaneously send the pages from the swap buffers to destination PerfD and CapD. That is, a page from CapD goes to PerfD through the first half of the doubled datapath and vice versa through the second half for another 128 tCK slots. As a result, we can reduce one iteration per migration process compared to the default implementation. In summary, a page swap in a conventional architecture requires 3 transactions, whereas it needs only 2 transactions in XPPath leveraging the doubled datapath.

Figure 7: Hot page migration process exploiting 3D-Path and XPath. 3D-Path can redirect a memory transaction from the occupied TSV channel to another free channel. XPath can (1) read data from CapD and PerfD to swap buffers on the logic die simultaneously and (2) write data back to destination dies concurrently by exploiting more TSVs of XPath.

Lastly, 3D-Path can also use XPath of CH[2] for ordinary memory or page-swap transactions. In this 3D-Path operation, there are two possible exceptions. First, the channel is not physically usable for the same bank group access because they share the inter-bank datapath. Second, a page-swap transaction is paused when there is no unused channel at a certain moment.

4.2 3D-XPath Management

Logic support for page swapping: To swap two pages between PerfD and CapD, we need control logic and support from memory controllers. The control logic can be implemented with simple enhancement of MBIST (Memory Built-In Self Test), which is implemented in recent industry 3D-stacked DRAM [35]. MBIST can generate commands and addresses in a Direct Memory Access (DMA) fashion. The memory controllers also need to know whether a requested page is located at PerfD or CapD. CapD and PerfD have different timing parameters, so we can use a slot-based interface which reserves a slot for slow memory responses on the datapath, similar to prior work [8]. This approach looks for a free slot and reserves the free slot as a backup slot when the page is located at CapD. We also set the transfer granularity of a page-swap transaction to 4 KB, the page size, based on a sensitivity study that shows the best performance at 4 KB.

Page remapping: It is desirable to swap a page in PerfD with any page in CapD, which is similar to a fully associative cache. However, this incurs a huge cost for a page remapping table. Instead, we assume that one page in PerfD can be swapped to one of a few pages in CapD, constituting a page group (similar to Congruence Group in [19]) which can be regarded as a rank. For example, with 2 PerfD and 6 CapD dies, each page group consists of 1 page from a PerfD die and 6 pages from 3 CapD dies (two from each). The pages within a page group are aligned such that they are placed at the same row address across banks, which significantly reduces the cost of remapping tables.

For a 3D-stacked DRAM with 28 GB of capacity, we need 2.125 MB of memory for the remapping table assuming 3 bits per 4 KB row. The logic die can provide sufficient space to hold this remapping table with SRAM. A more space efficient option is to use a cache and store the entire remapping table in PerfD. For example, a 24

KB remapping table cache achieves a hit rate of 94.1% on average for memory-intensive multi-programmed benchmarks (Section 6). When a memory request misses the remapping table cache, it brings the corresponding remapping table information from the reserved address space of PerfD. Our estimation using CACTI [17] with 32nm Low Standby Power (LSTP) technology shows that a 16-way 24 KB remapping table cache with 96 B per entry can translate a given page group (rank) address in 0.88 ns, which is less than 1 tCK slot³ and consumes 0.4 mm² (less than 1% area of a logic die).

To swap pages, we may follow the sequence described in Row-Clone [58]. That is, the standard 3D-stacked DRAM requires at least three times of row-copy latency (128 tCK slots plus activate/precharge overhead per 4 KB row copy) per page-swap transaction. However, XPath needs only two times of row-copying latency. Besides, we can pause a page-swap transaction by one ordinary memory transaction if a request needs to be sent to another bank group which is not involved with the page-swap transaction. The temporary row buffer can be located at a DRAM or logic die. If it is located at DRAM, a page-sized bulk copy mechanism such as LISA [16], can be applied to reduce page swapping latency between the temporary row buffer and the corresponding BLSAs. However, this requires one more row buffer per bank group of each DRAM. By contrast, when the buffer is located at the logic die, a page swap requires fewer buffers but more transactions. In this paper, we assume that a temporary row buffer per channel is placed on the logic die.

Interaction with memory requests from applications: During a page-swap transaction, a command for a memory request from applications can interrupt the page-swap transaction. That is, the page-swap transaction is paused since an ordinary memory transaction has a higher priority than a page-swap transaction. However, it is prohibited to precharge or update an opened (activated) row of any bank which is involved with a page swap transaction, because it requires activating the target row again or corrupts values in a page being swapped. To protect against these hazards, if a memory controller sends a command for ordinary memory requests to banks involved with a page-swap transaction, the stacked memory sends

³tCK is a DRAM clock cycle (1 ns in this study). A DRAM command spends 1 tCK slot.

an exception signal back to the memory controller through a dedicated pin (similar to AERR and DERR in HBM). Then, the memory controller suspends all commands which head to the bank. After the page-swap transaction is completed, the memory controller resumes transactions for these memory requests.

Hot-page selection algorithm: In any heterogeneous memory system, it is important to precisely determine which pages are hot because bringing wrong pages to fast memory degrades performance. To determine which pages to swap between PerfD and CapD, we devise a hot page selection algorithm. Specifically, we leverage CHOP-AFC (Caching HOT Pages Adaptive Filter Cache) [32], which was proposed for large DRAM cache, to place hot pages in PerfD without too many page swaps between CapD and PerfD.

CHOP is proposed to choose hot pages based on history counters. Similar to CHOP, our hot-page selection algorithm requires a history counter for each page, and the counter value (set to zero in the beginning) is increased by one when a request is sent to the corresponding page. When the count value of a page exceeds a configured threshold, that page is regarded as a hot page. This hot page will be swapped to PerfD, and the count values of a page group are right-shifted by 1 bit to attenuate the history information (the count values being reduced into half). As mentioned above, our algorithm requires counters for each page and locates them on the logic die of a 3D-stacked memory. For example, a 28 GB memory package which consists of two 2 GB PerfD dies and six 4 GB CapD dies, needs 5.25 MB SRAM storage for a 6 bit counter per 4 KB page. A logic die is large enough to hold that size [35]. The cost can be further reduced by implementing a counter cache in a way similar to populating the remapping table cache as described in Section 4.1. We estimate that the cost of an exemplar 48 KB counter cache located at the logic die is 0.8% based on CACTI [17].

Prior work [32] also proposed an enhanced version of CHOP (CHOP-AFC) to utilize memory bandwidth more efficiently. If the number of memory requests within a certain time period exceeds a threshold, CHOP-AFC transfers hot pages only. Otherwise, it swaps all accessed pages on demand. CHOP-AFC counts the number of memory requests, but it requires a significant amount of logic on the memory controller. By contrast, we monitor the number of pending memory requests in the request queue; such a feature is already implemented in the contemporary memory controller. When the queue size exceeds a given threshold value, the memory controller can send a command or a signal through a dedicated pin to a stacked memory.

High speed in-memory copy: With 3D-XPath, we can also perform any to any in-stack copy with no restriction of source and target addresses. The copy method can utilize the implemented migration hardware and bypass processors. It first activates the source row, saves data in the migration buffer in the logic die, and then copies data to the target row. When the source and destination channels to be copied are different, using 3D-Path implementation, the copy operation is performed without size and address restrictions. The user can control in-memory copy by sending an instruction to the corresponding memory controller. The time taken for copying is two 128 tCK cycles per 4 KB row copy in addition to time for activation of the source and target rows.

Table 1: Timing parameters of DRAM devices based on [27] and Section 3.2.

Symbol	Description	PerfD	CapD
tRCD	ACT to RD delay	12 ns	18 ns
tRP	PRE command period	13 ns	20 ns
tCL	RD to first data delay	12 ns	16 ns
tWR	WR recovery delay	16 ns	18 ns

5 EXPERIMENTAL METHODOLOGY

We modeled a chip-multiprocessor system to evaluate the system-level impact of the proposed stacked memory architecture. We modified dist-gem5 [7] to simulate full-system configurations with the network subsystem and McSimA+ [6] to support the asymmetric timing parameters for stacked memory, which are enumerated in Table 1. The system has 16 out-of-order cores. Each core operates at 4 GHz, has peak IPC of 4, and is equipped with separate L1 instruction and data caches and a combined L2 cache, all with 64B cache lines. The size and associativity of each L1 cache and L2 cache are 16 KB and 4, and 512 KB and 16, respectively. A linear hardware prefetcher [33] detects and prefetches streams of consecutive memory accesses. Each memory controller (MC) controls one proposed 3D-stacked memory with four 32 GB/s channels, and has 64 request queue entries. The capacity of our stacked memory is 28 GB. Each channel consists of 4 ranks and each rank consists of 4 bank groups. For PerfD dies each bank group consists of 8 banks, whereas for CapD dies each bank group consists of 16 banks. The memory controller adopts the Parallelism-AwaRe Batch Scheduling (PAR-BS) [46] and adaptive open page management policy [25]. For dist-gem5 simulations, we set the network and storage subsystem parameters to model a 10Gb Network Interface Card and high-performance SATA SSD.

We used two different OLDD applications, Apache and memcached [4]. For Apache, a server system is set with MySQL [5] and Apache2 [3] web server. We prepared Apache with different memory capacity values to show that the increase in memory capacity with CapD affects the performance. We sent enough queries to load the data from the storage into the main memory for these two applications for warm-up; then we made checkpoints and the client sent queries to determine the response time and service rates of the server. When the server receives a query through the Apache2 server, it finds and returns values from the database to the client.

To evaluate the performance of non-OLDD applications, we used four types of multi-programmed (mixed) workloads which consist of SPEC CPU2006 [20] applications executed with reference data sets. We used Simpoints [59] to extract the most representative simulation points of each SPEC CPU2006 application. Each simulation point consists of 100M instructions. For each multi-programmed workload, a simulation point is assigned to each core, and one or two highest weight points are used per application. We classified each benchmark to one of three groups based on the L2-cache Misses Per Kilo-Instructions (MPKI) [26], each called mix-high, mix-med, and mix-low (see Table 2). We populated six multi-programmed workloads from mix-high, called mix-high[0–5]. The other group, called mix-blend, consists of five applications

Figure 8: Memory capacity versus performance in terms of Responded Requests Per Second (RRPS) on Apache for “He”terogeneous 3D-stacked DRAM (He), He with “S”wapping pages (HeS), HeS with “X”Path (HeSX), and HeS with “3D-X”Path (HeS3D-X).

from mix-high, six from mix-med, and five from mix-low. We used SPLASH-2 [66] and PARSEC [13] benchmark suites to evaluate the performance of multi-threaded environment for regions of interest. We used the datasets in [12] for SPLASH-2, and simlarge datasets for PARSEC [13]. We used iperf [2] to measure the performance of the any-to-any in-stack copy proposed in this paper. iperf is a TCP/UDP-based network bandwidth measurement application that prints network bandwidth of the system. The major network overhead is interrupt processing cost, device driver overhead, check-summing, and buffer copying (the overhead of buffer copying is about 23%) [11]. Exploiting DMA reduces CPU overhead for memory copy, but it still suffers from throughput limitation originating from CPU-side datapath [50]. To show how faster memory copy affects network performance, we compared default C library memory copy, zero-copy using DMA, and 3D-XPath copy.

Table 1 shows the latency values of PerfD and CapD used for evaluation. The write recovery delay (tWR) of CapD is not significantly worse than that of default DRAM dies even if CapD employs In-DRAM ECC, as opposed to the prior work [15]. This is because the granularity of data reads and writes to an activated row (*i.e.*, 512b) is larger than the data size of a codeword of In-DRAM ECC (*e.g.*, 128b) because only a single DRAM die is involved in our 3D-stacked memory whereas several DRAM dies participate in data transfers for memory modules in DDRx. The page-swap scheme requires 1.8% more space for remapping table and hot page counter cache on the logic die. In addition, more TSVs and structure for XPath and 3D-Path cost 2% and 1.5% more space, respectively.

6 EVALUATION

To evaluate the effectiveness of 3D-XPath in this paper, we take the 3D-stacked **H**eterogeneous DRAM consisting of 2 PerfD and 6 CapD dies without 3D-XPath (denoted by He) as our baseline for performance evaluation, unless mentioned otherwise. We test the three other configurations as follows: (1) He with **S**wapping hot pages (HeS), (2) HeS with **X**Path (HeSX) and (3) HeS with **3D-X**Path (HeS3D-X).

6.1 OLDI Workloads

In this evaluation, we set the dataset size of Apache and memcached databases to 4 GB and 2 GB, respectively. We use such memory

Figure 9: Relative mean and 95th-percentile response times of Apache and memcached for He, HeS, HeSX, and HeS3D-X, where values are normalized to the mean response time of 4 GB He.

capacity considering that the full-system simulation time is proportional to the dataset size. However, to appropriately capture the impact of memory capacity on mean and 95th-percentile response times for a given dataset size, we first use a physical machine with larger memory capacity and dataset size. Then we configure our simulation environment such that the simulation gives similar trends in terms of mean and 95th-percentile response times for the chosen dataset size (*e.g.*, 4 GB for Apache). The evaluation of Apache was conducted in two dimensions.

First, we make clients send more Requests Per Second (RPS) than a server can handle, and measure the throughput of a given system in terms of Responded RPS (RRPS), as we change the memory capacity. In this scenario, RRPS is always smaller than RPS as some requests are not responded. When the dataset size is larger than the memory capacity, page faults (*i.e.*, page swaps between storage and memory) frequently occur, limiting RRPS. For example, Figure 8 shows that 4 GB He gives 78% more RRPS than 2 GB He, which shows the benefit of larger memory capacity. Furthermore, for the same memory capacity, HeS3D-X performs 24% (1 GB), 23% (2 GB), and 17% (4 GB) better than He. Considering the same number of stacked dies per DRAM package, 3D-XPath DRAM can provide 1.75× larger capacity than 3D-stacked DRAM with only PerfD. Although not shown in Figure 8, 14 GB HeS3D-X can provide 11% more RRPS than 8 GB 3D-stacked DRAM with only PerfD, demonstrating the benefit of a larger capacity heterogeneous memory system.

Second, to fairly compare mean and 95th-percentile response times across different configurations, we find the RPS that allows a server with HeS3D-X to respond to all the requests (*i.e.*, RRPS = RPS) and then apply that RPS to all other configurations. Figure 9 shows the mean and 95th-percentile response times for various configurations. Especially, we use uniform and skewed distributions to model the locality of service requests from clients. For the uniform distribution case, the mean and 95th-percentile response times decrease as we increase memory capacity (“apache-uniform” in Figure 9). For example, 8 GB He gives 62% and 66% lower mean and 95th-percentile response times than 4 GB He, respectively. For the skewed distribution case which exhibits the locality in requested items, 8 GB HeS3D-X offers 29%, 32%, and 23% lower 95th-percentile response time than 8 GB He, HeS, and HeSX, respectively (“apache-skewed” in Figure 9).

Table 2: SPEC CPU2006 multi-programmed workload groups.

Group	SPEC CPU2006 applications
mix-high[0–5]	mcf, milc, leslie3d, soplex, GemsFDTD, libquantum, lbm, omnetpp, sphinx3
mix-med	bzip2, gcc, bwaves, zeusmp, gromacs, cactusADM, h264ref, astar, wrf, xalancbmk
mix-low	perlbench, gamess, namd, gobmk, dealII, povray, calculix, hmmer, sjeng, tonto
mix-blend	mcf, bwaves, bzip2, calculix, lbm, milc, cactusADM, wrf, hmmer, soplex, xalancbmk, dealII, leslie3d, libquantum, h264ref, zeusmp

Figure 10: Relative IPC, EDP and average memory latency to handle LLC misses for (a) multi-programmed and (b) multi-threaded workloads. **Figure 11: Performance comparison over different migration granularity on mix-high0–3 with 4 KB DRAM page size.**

HeS actually gives 3% longer 95th-percentile response time than He because page-swap transactions block channels from servicing memory requests from Apache. Furthermore, although not shown in Figure 9, compared with 16 GB 3D-stacked DRAM with only PerfD, 28 GB HeS3D-X gives 3% lower 95th-percentile response time, but 28 GB He provides 2% longer 95th-percentile response time. These demonstrate the importance and efficacy of providing larger capacity with 3D-XPath. Figure 9 also shows the mean and 95th-percentile response times of memcached for 4 GB and 8 GB configurations. For memcached, which stores all data in main memory, 8 GB HeS3D-X offers 30%, 32%, and 25% lower 95th-percentile response time than 8 GB He, HeS, and HeSX, respectively.

6.2 Non-OLDI Workloads

Multi-programmed: For the most memory-intensive multi-programmed workload, mix-high0, HeS3D-X provides 4.9%, 3.2%, 2.0% higher performance and 5.2%, 4.5%, 3.4% lower read latency than He, HeS, and HeSX, respectively. This shows the benefit of 3D-XPath which reduces the negative impact of additional queuing delay imposed by more frequent accesses to PerfD. Evaluating all six mix-high workloads (Figure 10), HeS3D-X provides 2.8%, 1.9%, 1.0% higher performance and 3.4%, 3.3%, 1.6% lower read latency than He, HeS, and HeSX. As expected, mix-low workloads do not benefit

from HeS3D-X in terms of performance and read latency because they are less memory-intensive.

Multi-threaded: For memory-intensive radix, HeS, HeSX, and HeS3D-X give 1.1%, 2.6%, and 5.9% higher performance than the baseline He, respectively. radix has a high degree of locality in memory accesses, and hence utilizes the lower access latency offered by PerfD dies effectively. For other workloads, HeS3D-X provides up to 2.6% higher performance than He.

Memory-copy bandwidth: We ran iperf over various TCP window sizes to compare the performance of memory-copy. We tested three configurations: 1) using standard C library’s memory copy (normal), 2) exploiting DMA feature implemented at iperf (DMA), and 3) leveraging the copy instruction proposed at 3D-XPath (3D-X). With the proposed 3D-XPath copy, we achieve higher network bandwidth regardless of TCP window size. DMA can improve performance by offloading memory copying, but its degree of improvement is still limited. When TCP window size is 416 KB (the default maximum TCP window size of Linux), proposed 3D-X copy provides 39.8% higher network bandwidth than the normal copy.

6.3 Sensitivity Analysis: Migration Granularity

Figure 11 shows how different migration block sizes affect performance. On mix-high1, 4 KB migration block size shows the best performance for all. However, for the other workloads, performance is not much sensitive to migration block size. Thus, similar to the page size, we use a 4 KB page size because smaller granularity needs larger remapping tables and hot-page swap counters.

7 RELATED WORK

Enhancing I/O performance of 3D-stacked DRAM: Lee *et al.* modified the TSV structure on 3D-stacked DRAM for simultaneous multi-die accesses [38]. As opposed to 3D-XPath, they assumed

much higher bandwidth between dies and proposed that each die has different I/O bit rates.

Asymmetric DRAM: CHARM changed the aspect ratio of a DRAM mat, a small two-dimensional array of cells, to reduce access latency of DRAM banks depending on physical distance between a bank and I/O [65]. TL-DRAM [39] divides a bitline within a mat into two and provides a way to access a portion of cells closer to the corresponding sense amplifiers with lower latency. Shin *et al.* observed that newly refreshed row has more charges and proposed to access them with lower latency [60]. These are orthogonal to 3D-XPath and can be applied synergistically.

NVM+DRAM hybrid memory systems: JEDEC introduced a NVDIMM standard [24], where a DIMM combines DRAM and NVM where NVM preserves data of DRAM (NVDIMM-N) or composes a larger memory space with DRAM (NVDIMM-P). 3D-XPath can provide a high-performance memory-controller-agnostic solution to NVDIMM-P types of memory. PCM has been proposed as a sole replacement of DRAM for main memory [37] or combined with a last-level DRAM cache [44, 53]. They either adjusted row buffer size to alleviate the power impact of PCM writes or populated more row buffers to mitigate the performance penalty of high PCM latency. Dhiman *et al.* proposed another hybrid system where DRAM and PCM constitute separate partitions in a single address space [21], and they augmented memory controllers with access map cache and page swapping manager to reduce power consumption. Meza *et al.* proposed fine-granularity management for DRAM cache [44]. They proposed TIMBER for caching metadata for recently accessed rows in a buffer, with storing tags in memory. They replaced a large SRAM tag memory into a small SRAM tag cache. We used a small sized translation SRAM cache to achieve similar benefits.

Page placement in hybrid memory systems: Dong *et al.* migrate hot pages from an off-chip memory to an on-chip memory [22]. Through modifying on-chip memory controllers, it can manage migration between off-chip and on-chip memory. They focused on how to utilize on-chip and off-chip DRAM with migration whereas 3D-XPath is between hybrid off-chip memory types. [55] architected a memory controller to observe access patterns and to migrate page frames from PCM to DRAM. They proposed migration policies and augmented memory controllers to monitor access patterns whereas our paper suggested an alternative path and room for migration on stacked hybrid memory with built-in swap control logic. CAMEO [19], ToR [61], and SILC-FM [56] suggested mechanisms to swap data between fast and slow memory at the granularities of cache block (CAMEO), page (ToR), and in-betweens (SILC-FM). Compared to these proposals, 3D-XPath focused on optimizing the microarchitecture of fast and slow memory being stacked together to provide cost-effective alternative path between heterogeneous memory dies.

8 CONCLUSION

In this paper, we have proposed 3D-XPath, a managed DRAM architecture which provides cost-effective alternative paths for memory transactions on heterogeneous 3D-stacked memory composed of high-density and fast DRAM dies. 3D-XPath consists of 3D-Path, which diverts memory requests to a more lightly utilized channel

to mitigate a surge in access time due to a burst of data transfers, and XPath, which populates wider (doubled) datapath only for fast DRAM dies and dedicates swap buffers to cost-effectively swap pages between high-density and fast DRAM dies. Evaluating memory- and I/O-intensive applications where memory capacity, latency, and bandwidth all matter, we showed that 3D-XPath DRAM reduces the high-percentile response time of latency-sensitive applications by ~30% and improve throughput by ~39%, respectively, compared with DRAM without 3D-XPath.

ACKNOWLEDGMENTS

This work was supported in part by Samsung Electronics, NSF under Grant CNS-1705047 and CNS-1557244, Next-Generation Information Computing Development Program through NRF funded by the Ministry of Science, ICT & Future Planning (2015M3C4A7065647), and the NRF of Korea grant funded by the Korea government (NRF-2017R1A2B2005416).

REFERENCES

- [1] 2016. SK Hynix to Push its DRAM Technology as Next Global Standards. <http://www.ipnoms.net/?p=15826>
- [2] 2017. iPerf - The ultimate speed test tool for TCP, UDP and SCTP. <https://iperf.fr/>
- [3] 2018. Apache http server project. <https://httpd.apache.org/>
- [4] 2018. memcached - a distributed memory object caching system. <https://memcached.org/>
- [5] 2018. MySQL: The world's most popular open source database. <https://httpd.apache.org/>
- [6] J. Ahn, S. Li, S. O., and N. P. Jouppi. 2013. McSimA+: A Manycore Simulator with Application-level+ Simulation and Detailed Microarchitecture Modeling. In *ISPASS*.
- [7] M. Alian, G. Dozza, U. Darbaz, S. Diestelhorst, D. Kim, and N. S. Kim. 2017. dist-gem5: Distributed Simulation of Computer Clusters. In *ISPASS*.
- [8] R. B. Aniruddha N. Udipi, Naveen Muralimanohar. 2011. Combining Memory and a Controller with Photonics through 3D-Stacking to Enable Scalable and Energy-Efficient Systems. In *ISCA*.
- [9] H. Asghari-Moghaddam, Y. H. Son, J. Ahn, and N. S. Kim. 2016. Chameleon: Versatile and Practical Near-DRAM Acceleration Architecture for Large Memory Systems. In *MICRO*.
- [10] G. Ayers, J. Ahn, C. Kozyrakis, and P. Ranganathan. 2018. Memory Hierarchy for Web Search. In *HPCA*.
- [11] S. Bhattacharya and V. Apte. 2006. A measurement study of the Linux TCP/IP stack performance and scalability on SMP systems. In *Communication System Software and Middleware*.
- [12] C. Bienia, S. Kumar, and K. Li. 2008. PARSEC vs. SPLASH-2: A Quantitative Comparison of Two Multithreaded Benchmark Suites on Chip-Multiprocessors. In *IEEE International Symposium on Workload Characterization (IISWC)*.
- [13] C. Bienia, S. Kumar, J. P. Singh, and K. Li. 2008. The PARSEC Benchmark Suite: Characterization and Architectural Implications. In *PACT*.
- [14] R. Callaghan. 2014. ULLtraDIMM SSD Overview.
- [15] S. Cha, S. O., H. Shin, S. Hwang, K. Park, S. J. Jang, J. S. Choi, G. Y. Jin, Y. H. Son, H. Cho, J. H. Ahn, and N. S. Kim. 2017. Defect Analysis and Cost-effective Resilience Architecture for Future DRAM Devices. In *HPCA*.
- [16] K. K. Chang, P. J. Nair, D. Lee, S. Ghose, M. K. Qureshi, and O. Mutlu. 2016. Low-Cost Inter-Linked Subarrays (LISA): Enabling Fast Inter-Subarray Data Movement in DRAM. In *HPCA*.
- [17] K. Chen, S. Li, N. Muralimanohar, J. Ahn, J. B. Brockman, and N. P. Jouppi. 2012. CACTI-3DD: Architecture-level Modeling for 3D Die-stacked DRAM Main Memory. In *DATE*.
- [18] J. Choi. 2014. Next Big Thing: DDR4 3DS. In *Server Forum*.
- [19] C. Chou, A. Jaleel, and M. K. Qureshi. 2014. CAMEO: A Two-Level Memory Organization with Capacity of Main Memory and Flexibility of Hardware-Managed Cache. In *MICRO*.
- [20] S. P. E. Corporation. 2006. SPEC CPU2006. <https://www.spec.org/cpu2006/>
- [21] G. Dhiman, R. Ayoub, and T. Rosing. 2009. PDRAM: a Hybrid PRAM and DRAM Main Memory System. In *DAC*.
- [22] X. Dong, Y. Xie, N. Muralimanohar, and N. P. Jouppi. 2010. Simple but Effective Heterogeneous Main Memory with On-Chip Memory Controller Support. In *SC*.
- [23] X. Dong, J. Zhao, and Y. Xie. 2010. Fabrication Cost Analysis and Cost-Aware Design Space Exploration for 3-D ICs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 29, 12 (Dec 2010), 1959–1972.

- [24] B. Gervasi. 2016. NVDIMM-P: A New Hybrid Architecture. In *Open Server Summit (OSS)*.
- [25] Intel. 2010. *Intel Xeon Processor 7500 Series Datasheet*.
- [26] A. Jaleel. 2010. Memory Characterization of Workloads Using Instrumentation-Driven Simulation. *Web Copy: <http://www.glue.umd.edu/ajaleel/workload>* (2010).
- [27] JEDEC Standard. 2015. High Bandwidth Memory (HBM) DRAM. *JEDEC235A* (2015).
- [28] JEDEC Standard. 2016. DDR4 SDRAM Load Reduced DIMM Design Specification. *JEDEC21-C* (2016).
- [29] JEDEC Standard. 2016. DDR4 SDRAM Registered DIMM Design Specification. *JEDEC21-C* (2016).
- [30] JEDEC Standard. 2016. Graphics Double Data Rate (GDDR5X) SGRAM Standard. *JEDEC232A* (2016).
- [31] JEDEC Standard. 2017. LOW POWER DOUBLE DATA RATE 4X (LPDDR4X). *JEDEC209-4-1* (2017).
- [32] X. Jiang, N. Madan, L. Zhao, M. Upton, R. Iyer, S. Makineni, D. Newell, Y. Solihin, and R. Balasubramanian. 2011. CHOP: Integrating DRAM Caches for CMP Server Platforms. *IEEE micro* (2011).
- [33] N. P. Jouppi. 1990. Improving Direct-Mapped Cache Performance by the Addition of a Small Fully-Associative Cache and Prefetch Buffers. In *ISCA*.
- [34] H. Kasture and D. Sanchez. 2016. Tailbench: a Benchmark Suite and Evaluation Methodology for Latency-critical Applications. In *IISWC*.
- [35] J. Kim and Y. Kim. 2014. HBM: Memory Solution for Bandwidth-Hungry Processors. In *Hot Chips*.
- [36] J. Kim, M. Sullivan, and M. Erez. 2015. Bamboo ECC: Strong, Safe, and Flexible Codes for Reliable Computer Memory. In *HPCA*.
- [37] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger. 2009. Architecting Phase Change Memory as a Scalable DRAM Alternative. In *ISCA*.
- [38] D. Lee, S. Ghose, G. Pekhimenko, S. Khan, and O. Mutlu. 2016. Simultaneous Multi-Layer Access: Improving 3D-Stacked Memory Bandwidth at Low Cost. *ACM TACO* (January 2016).
- [39] D. Lee, Y. Kim, V. Seshadri, J. Liu, L. Subramanian, and O. Mutlu. 2013. Tiered-Latency DRAM: A Low Latency and Low Cost DRAM Architecture. In *HPCA*.
- [40] H. G. Lee, S. Baek, C. Nicopoulos, and J. Kim. 2011. An Energy- and Performance-Aware DRAM Cache Architecture for Hybrid DRAM/PCM Main Memory Systems. In *Intl. Conf. on Computer Design (ICCD)*.
- [41] S. Lee, H. Cho, Y. H. Son, Y. Ro, N. S. Kim, and J. Ahn. 2018. Leveraging Power-Performance Relationship of Energy-Efficient Modern DRAM Devices. *IEEE Access* 6 (2018).
- [42] S.-H. Lee. 2016. Technology scaling challenges and opportunities of memory devices. In *IEEE International Electron Devices Meeting (IEDM)*.
- [43] G. H. Loh and M. D. Hill. 2011. Efficiently Enabling Conventional Block Sizes for Very Large Die-stacked DRAM Caches. In *MICRO*.
- [44] J. Meza, J. Chang, Y. HanBin, O. Mutlu, and P. Ranganathan. 2012. Enabling Efficient and Scalable Hybrid Memories Using Fine-Granularity DRAM Cache Management. In *IEEE Computer Architecture Letters*.
- [45] Micron. 2011. RLD3RAM3 Datasheet.
- [46] O. Mutlu and T. Moscibroda. 2008. Parallelism-Aware Batch Scheduling: Enhancing Both Performance and Fairness of Shared DRAM Systems. In *ISCA*.
- [47] P. J. Nair, D.-H. Kim, and M. K. Qureshi. 2013. ArchShield: Architectural Framework for Assisting DRAM Scaling by Tolerating High Error Rates. In *ISCA*.
- [48] P. J. Nair, V. Sridharan, and M. K. Qureshi. 2016. XED: Exposing On-Die Error Detection Information for Strong Memory Reliability. In *ISCA*.
- [49] R. Oh, B. Lee, S.-W. Shin, W. Bae, H. Choi, I. Song, Y.-S. Lee, J.-H. Choi, C.-W. Kim, S.-J. Jang, and J. S. Choi. 2014. Design Technologies for a 1.2V 2.4Gb/s/pin High Capacity DDR4 SDRAM with TSVs. In *VLSI Circuits Digest of Technical Papers*.
- [50] S. K. Palaniappan and P. B. Nagaraja. 2008. Efficient data transfer through zero copy. <https://www.ibm.com/developerworks/library/j-zero-copy/>
- [51] J. Park, Y. Hwang, S.-W. Kim, S. Han, J. Park, J. Kim, J. Seo, and B. Kim. 2015. 20nm DRAM: A new beginning of another revolution. In *IEDM*.
- [52] J. T. Pawlowski. 2011. Hybrid Memory Cube (HMC). In *Hot Chips*.
- [53] M. K. Qureshi, V. Srinivasan, and J. A. Rivers. 2009. Scalable High Performance Main Memory System Using Phase-Change Memory Technology. In *ISCA*.
- [54] R. K. Ramanujan, R. Agarwal, and G. J. Hinton. 2011. Apparatus and Method for Implementing a Multi-level Memory Hierarchy Having Different Operating Modes. US Patent App. 13/994,731.
- [55] L. Ramos, E. Gorbato, and R. Bianchini. 2011. Page Placement in Hybrid Memory Systems. In *ICS*.
- [56] J. H. Ryoo, M. R. Meswani, A. Prodromou, and L. K. John. 2017. SILC-FM: Subblocked InterLeaved Cache-Like Flat Memory Organization. In *HPCA*.
- [57] Samsung Semiconductor. 2016. Research collaboration communications.
- [58] V. Seshadri, Y. Kim, C. Fallin, D. Lee, R. Ausavarungnirun, G. Pekhimenko, Y. Luo, O. Mutlu, P. B. Gibbons, M. A. Kozuch, and T. C. Mowry. 2013. RowClone: Fast and Energy-Efficient In-DRAM Bulk Data Copy and Initialization. In *MICRO*.
- [59] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder. 2002. Automatically Characterizing Large Scale Program Behavior. In *ASPLOS*.
- [60] W. Shin, J. Yang, J. Choi, and L.-S. Kim. 2014. NUAT: A Non-Uniform Access Time Memory Controller. In *HPCA*.
- [61] J. Sim, A. R. Alameldeen, Z. Chishti, C. Wilkerson, and H. Kim. 2014. Transparent Hardware Management of Stacked DRAM as Part of Memory. In *MICRO*.
- [62] S. Sivaram. 2016. Storage Class Memory: Learning from 3D NAND.
- [63] A. Sodani. 2015. Knights Landing (KNL): 2nd Generation Intel® Xeon Phi Processor. In *Hot Chips*.
- [64] Y. H. Son, S. Lee, S. O, S. Kwo, N. S. Kim, and J. Ahn. 2015. CiDRA: A Cache-inspired DRAM Resilience Architecture. In *HPCA*.
- [65] Y. H. Son, S. O, Y. Ro, J. W. Lee, and J. Ahn. 2013. Reducing Memory Access Latency with Asymmetric DRAM Bank Organizations. In *ISCA*.
- [66] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta. 1995. The SPLASH-2 Programs: Characterization and Methodological Considerations. In *ISCA*.
- [67] C. Xu, D. Niu, N. Muralimanohar, R. Balasubramanian, T. Zhang, S. Yu, and Y. Xie. 2015. Overcoming the Challenges of Crossbar Resistive Memory Architectures. In *HPCA*.