

# Simultaneous Multi Layer Access: A High Bandwidth and Low Cost 3D-Stacked Memory Interface

Donghyuk Lee      Gennady Pekhimenko  
donghyul@cmu.edu    gpekhime@cs.cmu.edu  
Samira Khan      Saugata Ghose      Onur Mutlu  
samirakhan@cmu.edu    ghose@cmu.edu    onur@cmu.edu

Carnegie Mellon University

SAFARI Technical Report No. 2015-008

June 8, 2015

## Abstract

Limited memory bandwidth is a critical bottleneck in modern systems. 3D-stacked DRAM enables higher bandwidth by leveraging wider Through-Silicon-Via (TSV) channels, but today's systems cannot fully exploit them due to the limited internal bandwidth of DRAM. DRAM reads a whole row simultaneously from the cell array to a row buffer, but can transfer only a fraction of the data from the row buffer to peripheral IO circuit, through a limited and expensive set of wires referred to as global bitlines. In presence of wider memory channels, the major bottleneck becomes the limited data transfer capacity through these global bitlines. Our goal in this work is to enable higher bandwidth in 3D-stacked DRAM without the increased cost of adding more global bitlines. We instead exploit otherwise-idle resources, such as global bitlines, already existing within the multiple DRAM layers by accessing the layers simultaneously. Our architecture, Simultaneous Multi Layer Access (SMLA), provides higher bandwidth by aggregating the internal bandwidth of multiple layers and transferring the available data at a higher IO frequency.

To implement SMLA, simultaneous data transfer from multiple layers through the same IO TSVs requires coordination between layers to avoid channel conflict. We first study coordination by static partitioning, which we call Dedicated-IO, that assigns groups of TSVs to each layer. We then provide a simple, yet sophisticated mechanism, called Cascaded-IO, which enables simultaneous access to each layer by time-multiplexing the IOs, while also reducing design effort. By operating at a frequency proportional to the number of layers, SMLA provides a higher bandwidth (4X for a four-layer stacked DRAM). Our evaluations show that SMLA provides significant performance improvement and energy reduction (55%/18% on average for multi-programmed workloads, respectively) over a baseline 3D-stacked DRAM with very low area overhead.

## 1 Introduction

Main memory, predominantly built using DRAM, is a critical performance bottleneck in modern systems due to its limited bandwidth [2, 30]. With increasing core count and memory intensive applications, memory bandwidth is expected to become a greater constraint in the future [5]. For the last few decades, DRAM vendors provided higher bandwidth by using higher IO frequencies, increasing the bandwidth available per pin (a 17x improvement over the last decade, from 400Mbps in DDR2 to 7Gbps in GDDR5 [32]). However, further increase in frequency is challenging due to higher energy consumption and logical complexity. Recent development in 3D integration through *Through-Silicon Vias* (TSVs) provides an alternate way to provide higher bandwidth. TSVs enable wider IO interface among vertically stacked layers in 3D-stacked DRAM architectures [16, 13, 8, 14, 12, 18].

Even though the TSV technology enables higher data transfer capacity, existing 3D-stacked DRAMs cannot fully take advantage of the potential wide interface. For example, Wide-IO [16] offers an extremely high bus width (512 bit, 16 – 64 times wider than conventional DRAM chips), but can only operate at much lower frequencies (200 – 266MHz) than conventional DRAMs (e.g., as high as 2133MHz for DDR3). As a result, the effective bandwidth increase is an

order of magnitude lower than the bus width increase. Based on our analysis of the DRAM hierarchy (explained in Section 2), 3D-stacked DRAMs cannot fully exploit the wider IO interface due to their limited internal bandwidth. DRAM reads a whole row simultaneously from the cell array to a row buffer, but can transfer only a fraction of data in the row buffer through a limited set of wires (global bitlines) and sense amplifiers. Therefore, the number of global bitlines dominates the DRAM internal data transfer capacity. One naive way to enable higher bandwidth is to add more global bitlines and sense amplifiers [13, 12]. However, the area and energy constraints make this intuitive and simple solution extremely expensive.<sup>1</sup>

Our goal in this work is to enable higher bandwidth in 3D-stacked DRAM without effectively increasing the cost by adding extra global sense amplifiers. In order to design such a system, we make the observation that a large number of global bitline interfaces already exist across the multiple layers of a 3D-stacked DRAM, but only a fraction of them are being enabled at any particular point of time. We exploit these otherwise-idle interfaces to access multiple DRAM layers *simultaneously*, which can increase the data delivery bandwidth to the TSV-based IO interface (which already has enough bandwidth to serve this additional data). We call this architecture *Simultaneous Multi Layer Access (SMLA)*. Using SMLA, multiple global bitline interfaces in multiple layers provide data to IO, enabling the IO interface (that is vertically connected across all stacked layers) with enough data to warrant transfers at a much higher frequency than conventional 3D-stacked DRAM.

To implement the idea of SMLA, simultaneous data transfer from multiple layers through the same IOs requires co-ordination between layers to avoid channel contention. One simple way to enable an SMLA architecture is to restrict the sharing of channels and assign IOs dedicated to each layer. We refer to this simple solution as *Dedicated-IO*, where fractions of TSVs form an IO group that is dedicated to a single layer (Figure 1b). Each layer has access to a smaller number of IOs, but can transfer the same amount of data by operating its IOs at a higher frequency. As each layer transfers data simultaneously at a higher frequency, Dedicated-IO enables bandwidth proportional to the number of layers (4X for a four-layer stacked DRAM) than the baseline system (Figure 1a).<sup>2</sup> While this architecture enables higher bandwidth, it has two disadvantages. First, as each layer requires dedicated connection to specific channels, the design of each layer is not uniform anymore, resulting in higher manufacturing cost. Second, the IO clock frequency scales linearly with the number of layers, resulting in greater dynamic energy consumption.

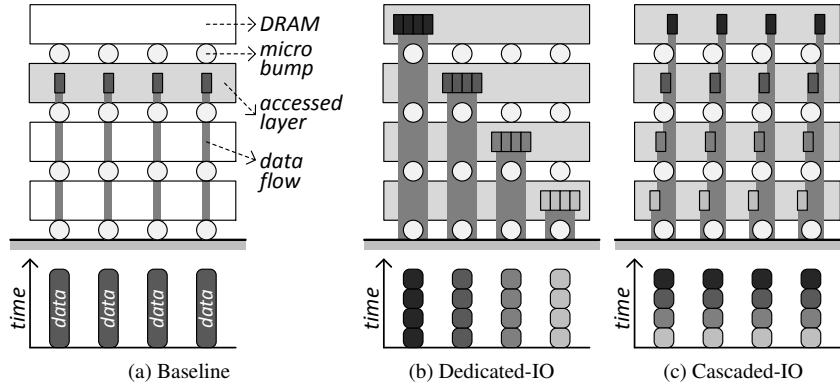


Figure 1: Single Layer (baseline) vs. Multi Layer Access

To solve these problems, we propose *Cascaded-IO*, which exploits the architectural organization of TSV-based interface, where an upper layer transfers its data through lower layers. Cascaded-IO (Figure 1c) enables simultaneous access to each layer by time-multiplexing the IOs. In this design, each layer first sends its own data and then sends data transferred from the upper layers. By operating at a frequency proportional to the number of layers, Cascaded-IO provides a much higher bandwidth (4X for a four-layer stacked DRAM). While Cascaded-IO operates at a higher frequency, we observe that only the bottom layer theoretically needs to operate at the highest frequency, as this is the only layer that transfers data from all upper layers. We propose to reduce the frequency of other layers and optimize the

<sup>1</sup>We estimate that global sense amplifiers (and their corresponding control logic) consume 5.18% of the total area in a 55nm DDR3 chip [37, 27]. Considering that 3D-stacked DRAM already contains many more global sense amplifiers (see Section 2.3), adding further sense amplifiers may be quite costly. The sense amplifier also consumes a large amount of energy because it is typically implemented as a differential amplifier, whose performance is strongly dependent on standby current [15, 28].

<sup>2</sup>A new design from Hynix [12] proposes to use exclusive IO interfaces for each layer of 3D-stacked DRAM, but enables higher bandwidth by increasing the internal bandwidth with extra global bitlines. Dedicated-IO, on the contrary, enables higher bandwidth at low-cost by leveraging existing global bitline interfaces and operating at a higher frequency (described in Section 9).

frequency individually for each layer based on the bandwidth requirements. As a result, Cascaded-IO enables higher bandwidth and lower energy consumption with a homogeneous DRAM design. While each layer in Cascaded-IO can operate at different frequency, we still expect that Cascaded-IO requires similar validation effort to existing DRAMs, as existing DRAMs guarantee correct operation at slower timing/frequency than their specification.

Our work makes the following contributions:

1. We propose a new 3D-stacked DRAM organization Simultaneous Multi Layer Access (SMLA), which enables higher bandwidth with low cost by leveraging the otherwise-idle interfaces in multiple layers of 3D-stacked memory.
2. We introduce mechanisms to transfer data from multiple layers without conflicting in the shared IO interface of 3D-stacked DRAMs. Simply assigning IOs to different layers (Dedicated-IO) avoids channel contention, but increases manufacturing cost and energy consumption. We propose an elegant and simple mechanism called Cascaded-IO, which time-multiplexes shared IOs so that each layer in 3D-stacked DRAM first transfers data from own layer and then from upper layers. Our mechanism has three major benefits. First, it provides bandwidth proportional to the number of layers in 3D-stacked DRAM with low cost. Second, it avoids non-uniformity in the design by time-multiplexing the IO interface, reducing manufacturing cost. Third, it enables higher bandwidth with lower energy consumption by optimizing IO frequency based on each layer's bandwidth requirement.
3. Our extensive evaluation of 3D-stacked DRAMs on 31 applications from SPEC CPU2006, TPC, and STREAM applications suites shows that our proposed mechanisms significantly improve performance and reduce energy consumption (55%/18% on average of 16-core multi-programmed workloads, respectively) over conventional 3D-stacked DRAMs (with no support for SMLA).

## 2 3D-Stacked DRAM Bandwidth Constraints

To understand the internal bandwidth bottleneck of 3D-stacked DRAM, we first describe the unique design of the IO interface in 3D-stacked DRAM (Section 2.1). Then, we walk through the DRAM architectural components that make up the datapath used for both read and write requests. We analyze the bandwidth of each component along this datapath in Section 2.2, and study a trade-off between bandwidth and area in Section 2.3.

### 2.1 Using TSVs to Increase IO Bandwidth

We first explain the detailed organization of a 3D-stacked DRAM that integrates a much wider IO bus than conventional DRAM. Figure 2a shows a 3D-stacked DRAM consisting of four layers that are connected over micro-bump and TSV interfaces. A TSV interface vertically connects all of the layers. The bottom of the four-stacked layer is either directly placed on top of a processor chip or connected to a processor chip by metal wires. Figure 2b details the organization of these TSV and micro-bump connections. Two metal lines connect each layer to these interfaces. The top metal line is connected to the micro-bump of its upper layer, and the bottom metal line is connected to the TSV. At the bottom of the chip, this TSV is exposed and connected to a micro-bump, connecting to the top metal line of the lower layer. These two metal lines are eventually connected by via or over peripheral circuits. In this way, several such layers can be stacked one on top of another.

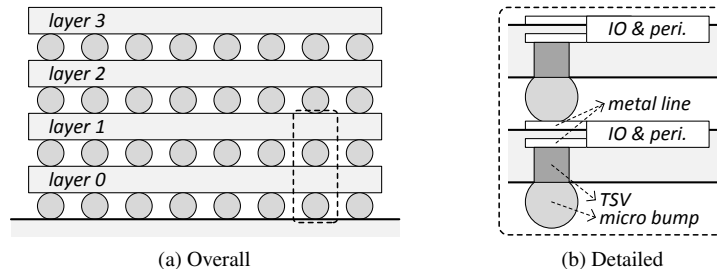


Figure 2: TSV Interface in 3D-Stacked DRAM

TSVs provide two major benefits to increase memory bandwidth and energy efficiency. First, due to the small feature size of modern TSV technologies (10 – 35 $\mu\text{m}$  pitch for TSVs [16, 38, 7, 6] vs. 90 $\mu\text{m}$  pitch for conventional pads), 3D-stacked DRAM can integrate hundreds of TSVs for its connections between layers. Second, the small capacitance of a TSV reduces energy consumption. Compared to conventional DRAM, whose off-chip IO are connected by long bonding wires and metal connections in the package, TSV connections between layers are very short, leading to lower capacitance and energy consumption during data transfer. As a result, 3D-stacked DRAM offers a promising DRAM architecture that can enable both higher bandwidth and energy efficiency.

## 2.2 DRAM Bandwidth Analysis

DRAM has two kinds of accesses (read and write) that transfer data through mostly the same path between its cells and off-chip IO. Figure 3a shows the read datapath from the DRAM cells to IO. When activating a row, all data in a row is transferred to a row buffer (a row of local sense amplifiers) through bitlines. Then, when issuing a read command with a column address, small fraction of data in the row buffer (corresponding to the issued column address) is read by the global sense amplifiers. Peripheral logic then transfers this data to the off-chip IO. To write data, commands containing data are issued, and peripheral logic transfers this data to the global sense amplifiers that write the data to a row buffer, which then writes the data to the cells corresponding to the requested address.

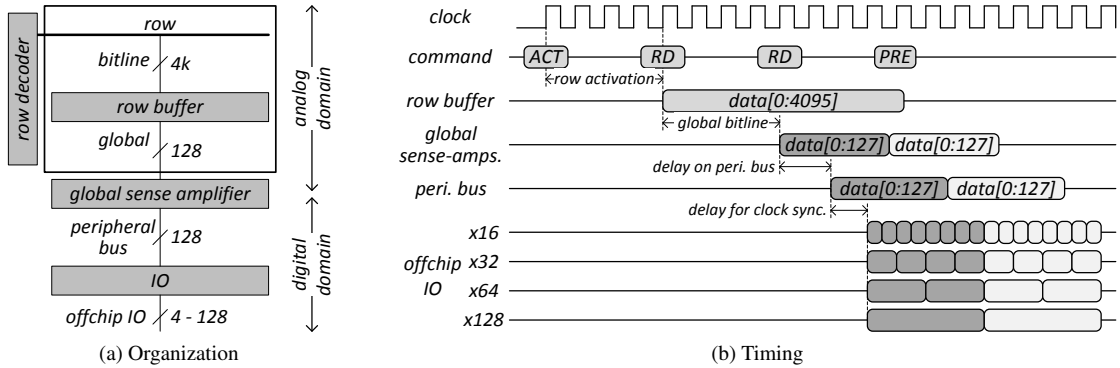


Figure 3: Overview of DRAM Organization

At each step, the individual structures along the datapath have their own data transfer rate (bandwidth). For a read access, we explore the bandwidth of each step in detail. When activating a row, all data within the row (e.g., 8Kbits [22]) is transferred to the row buffer, which takes about 13ns (based on  $t_{\text{RCD}}$  timing constraint [22]). Therefore, the bandwidth of the activation step is about 78.8GBps (Giga-Bytes-per-Second). After migrating data to a row buffer, global sense amplifiers read 64 – 128 bits of data from the row buffer for about 3 – 5ns through the global bitline.<sup>3</sup> Therefore, the bandwidth of the global bitline is about 2 – 4GBps. The data read by global sense amplifiers is transferred to the IO interface, which can operate at frequencies as high as 2GHz. Considering that the off-chip IO bus width can range from 4 to 128 bits, the off-chip bandwidth can be in the range of 1 – 32GBps.

Based on the above analysis, in conventional DRAMs that have small off-chip bus width (4 to 32 bits), the overall DRAM bandwidth can be bottlenecked by the bandwidth of the off-chip IO interface. However, with a wide-width off-chip IO interface (e.g., the 512-bit TSV interface on 3D-stacked DRAM), the global sense amplifiers turn into the dominant bottleneck of the overall DRAM bandwidth. Unfortunately, increasing the bandwidth of the global bitline interface requires large area overhead due to the need for additional global sense amplifiers and global bitlines (as we will discuss in Section 2.3). Furthermore, increasing the bandwidth of the global bitline interface requires a significant amount of additional energy. A differential amplifier, a popular primitive to implement the global sense amplifier, consumes significant levels of standby current. Due to the limited available global sense amplifier bandwidth, higher-frequency TSV interfaces for 3D-stacked DRAMs are currently unnecessary.

To show the relationship between the operating frequency and the bus width of the IO interface, we draw timing diagrams for the data bus during a read access (Figure 3b). We divide the DRAM hierarchy into two domains, (i) an analog domain, whose components consist of sensing structures and the data repository (the row buffer and global

<sup>3</sup>Due to the high latency of global bitline, DRAM operates at much lower frequency (200 – 266MHz) than its IO frequency [32]. Therefore, the ratio between IO frequency and in-DRAM frequency is same as the ratio between global bitline width and IO width, referred to as *Fetch Size* in DRAM.

sense amplifiers in Figure 3a), and (ii) a digital domain, whose components (the peripheral bus and off-chip IO in Figure 3a) transfer data with normal peripheral transistors and full swing of the data value (0 or 1).

In the analog domain, a data repository directly maps to a sensing structure with a fixed metal wire, and accessing this repository (the row buffer) from a global sense amplifier takes a fixed latency. During this global bitline latency, only one bit of data can be transferred over one global bitline. Therefore, the bandwidth of the global bitline interface is tightly coupled with the number of global bitlines and global sense amplifiers (128 bits in this example). Considering that each global sense amplifier is directly connected to a row buffer through a single wire and detects the intermediate voltage value (between 0 to  $V_{dd}$ ) of a row buffer, it is difficult to divide the global bitline interface into multiple stages to get the benefits of pipelining.

On the other hand, the peripheral interface (from the global sense amplifiers to off-chip IO) and IO interface (toward off-chip) can be divided into multiple stages and implemented in a pipelined manner. Therefore, as shown in Figure 3b, the 128 bits of data from global sense amplifiers can be transferred through either a narrow IO bus with higher frequency (e.g., DDR3 with a 16-bit bus) or wide IO bus with lower frequency (e.g., Wide-IO with a 128-bit bus per channel).<sup>4</sup> In the ideal case, assuming that the global bitline interface provides enough data to the peripheral interface, the wide bus in off-chip IO can transfer the data at higher frequency, enabling much higher bandwidth (i.e.,  $128\text{bits} \times 2\text{GHz} = 32\text{Gbps}$ ).

Based on this analysis, we observe that even though 3D-stacked DRAM can potentially enable much greater bandwidth, the limited bandwidth on the global bitline interface significantly limits the actually available bandwidth.

### 2.3 Bandwidth vs. Global Sense Amplifiers

Figure 4 shows the relationship between the bandwidth of DRAM-based memory systems (one chip for conventional DRAM, four layers for 3D-stacked DRAM) and the number of global sense amplifiers in a DRAM chip. We conservatively plot the minimum number of global sense amplifiers that achieves the proposed bandwidth in each design.<sup>5</sup>

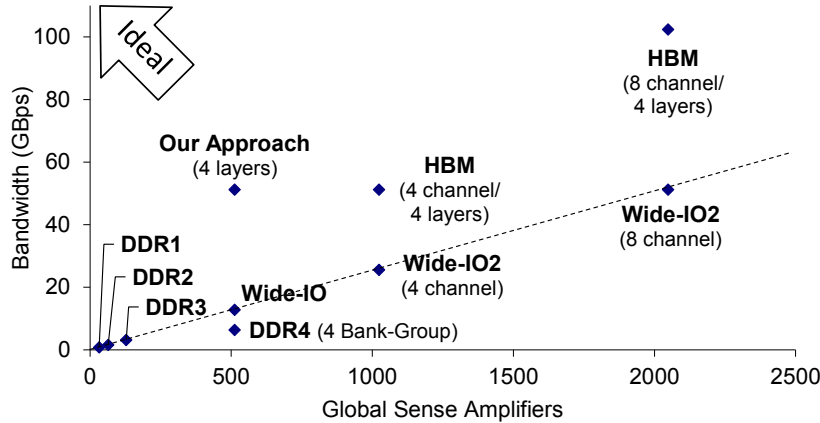


Figure 4: Bandwidth vs. Global Sense Amplifiers (Selecting DRAM bins that in-DRAM frequency is 200MHz)

Figure 4 shows that the bandwidth of DRAM mostly has been a linear function of the number of global sense amplifiers across several generations. In early DRAMs, the number of global sense amplifier structures was small (e.g., 64 for a DDR2 chip) and most likely was not the dominant source of DRAM area. Therefore, DRAM vendors had scaled the number of global sense amplifiers with IO frequency to increase the bandwidth. However, recently proposed 3D-stacked DRAMs [9, 13, 12] drastically increase the number of global sense amplifiers. For example, HBM [12, 18] offers the highest bandwidth, but requires a large number of global sense amplifiers (16X more than DDR3). This drastic increase comes from introducing TSV technologies that enable a wider, area-efficient bus. In the presence of the wider bus, the dominant bottleneck in achieving higher bandwidth now has shifted to the efficient implementation of global sense amplifiers.

<sup>4</sup>Conventional 3D-stacked DRAM consists of multiple channels that can operate independently – i.e, four channels of Wide-IO [16, 9], four or eight channels of Wide-IO2 [13] and HBM [12].

<sup>5</sup>For example, for DDR3 DRAM with an x16 IO bus and 8 times more global bitlines (x128) than IO bus, we assume that DDR3 has 128 global sense amplifiers (and 128 global bitlines), even though DDR3 can be designed to have multiple sets of global bitline interfaces, as this depends on the process technology and product design.

We estimate the area of global sense amplifiers (and their corresponding control logic) to be 5.18% of the total chip area in a 55nm DDR3 DRAM [37, 27]. Taking into account (i) the large area of global sense amplifiers in DDR3 and (ii) the recent increase in the number of sense-amplifiers in 3D-stacked DRAMs, we expect that adding more global sense amplifiers will become extremely expensive. As we explain in Section 3, in this paper, we enable much higher bandwidth at low-cost using the existing structure of 3D-stacked DRAM, approaching the ideal case (the top left corner of Figure 4).

### 3 Opportunities for Increasing Bandwidth

Even though 3D-stacked DRAM integrates a much wider IO bus through the use of TSVs, the limited internal bandwidth within the DRAM chip still constrains the total bandwidth that we can extract from this wider IO, as we analyzed in Section 2.2. While one solution may be to expand the number of global bitlines and sense amplifiers, this can be a costly prospect (as we show in Section 2.3). Instead, we would like to investigate orthogonal approaches to overcome the global bitline bottleneck, that exploit the existing structures more efficiently.

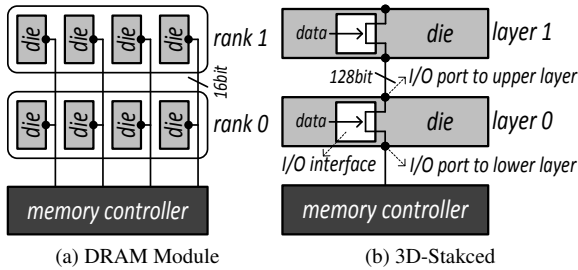


Figure 5: DRAM Module vs. 3D-DRAM

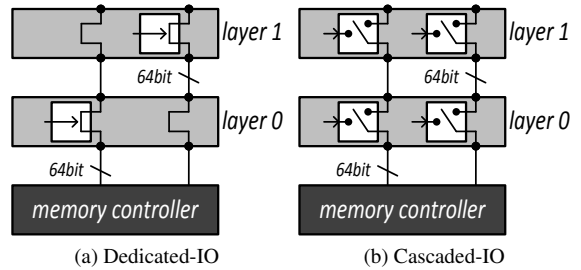


Figure 6: Proposals for More Bandwidth

We examine the architectural similarities and differences between 3D-stacked DRAM and conventional DRAM modules, and use this comparison to identify challenges and opportunities that are unique to the 3D-stacked DRAM design. We will use these observations to drive our low-cost approach. To aid with this comparison, we provide a side-by-side logical view of these two architectures in Figure 5, and focus on three major observations:

1. *Each layer in 3D-stacked DRAM acts as a rank, operating independently but sharing the IO interface with each other, which is similar to the rank organization of conventional DRAM modules.*

Figure 5a shows a DRAM module that has two ranks, each consists of four DRAM chips. All DRAM chips in a rank share control logic and operate in lockstep. The two ranks share the IO interfaces such that requests to different ranks are served serially. Figure 5b shows the organization of a 3D-stacked DRAM-based system that has two stacked layers. Similar to the multi-rank DRAM module, each layer in 3D-stacked DRAM can operate independently, but still shares the IO interface (TSV connection). Therefore, the TSV interface transfers data from different layers serially.

2. *Only one stacked chip forms a rank in 3D-stacked DRAM, compared to multiple chips in conventional DRAM module.*

Figure 5 shows that four conventional DRAM chips form a rank by dividing the 64-bit IO bus into four smaller 16-bit IO buses, with each smaller bus connected to a different chip. In contrast, each 3D-stacked DRAM layer is made up of a single chip, with the entire TSV bus connected to it.

3. *Each layer of 3D-stacked DRAM has two ports that can be easily decoupled by peripheral logic, whereas conventional DRAM has only a single port.*

As shown in Figure 2b, each stack layer in 3D-stacked DRAM is connected to its neighboring upper stack layer through the topmost metal layer. Each stack layer is also connected to its lower stack layer through one of its middle metal layers (different from the topmost metal layer). These two metal layers can be connected within the DRAM stack layer by either fixed via or peripheral logic. These two different metal layers form two independent IO ports, whose connectivity can be controlled by peripheral logic. In the general organization of 3D-stacked DRAM, the two metal wires that form the DRAM input ports are simply connected by via structures, meaning that they are connected but can be easily decoupled to support independent data transfers on each port.

As we described above, the major bandwidth bottleneck in 3D-stacked DRAM-based memory system is the global bitline interface in DRAM. Our approach to solve this problem is inspired by the organization of the conventional DRAM module, where multiple DRAM chips operate simultaneously, thereby increasing the overall bandwidth of the memory system (Figure 5a). From our observations, we conclude that while the 3D-stacked layers individually look different from the conventional DRAM architecture perspective, the ability to decouple input ports can allow us to treat multiple layers similarly to the way we treat multiple chips within DRAM.

## 4 Simultaneous Multi Layer Access

We propose *Simultaneous Multi Layer Access* (SMLA), a new mechanism that enables the concurrent use of multiple stack layers (chips) within 3D-stacked DRAM. SMLA exploits most of the existing DRAM structures within the chip, altering the IO logic to include multiplexing that supports these concurrent operations. This allows the bandwidth gains obtained by SMLA to be complementary to the gains that could be otherwise obtained from scaling these DRAM structures (e.g., adding more global bitlines and sense amplifiers).

In this section, we will describe two implementations for coordinating these multiplexed IOs. The first, *Dedicated-IO*, divides the TSV bus into several groups, with each group being dedicated to the IO of a single stack layer. The second, *Cascaded-IO*, explores the idea of time-multiplexing the bus instead of partitioning it, by exploiting our observation that we can decouple the input ports to in effect vertically segment the TSVs (Section 3). As we will discuss in Section 4.2, the Cascaded-IO implementation allows the circuit design of each stack layer to be identical, reducing design effort, and offers a more energy-efficient approach to implement SMLA.

### 4.1 Dedicated-IO

The Dedicated-IO implementation divides the wide TSV bus into several narrower groups, each of which is dedicated to a single stack layer (a DRAM chip). As Figure 6a shows, in the case of a two layer 3D-stacked DRAM, each layer has its own IO connection, whose width is half of the total IO width  $W$  (128 bits in the figure). With  $L$  layers working simultaneously, since each still contains  $W$  global bitlines and sense amplifiers, we can now transmit  $W \times L$  bits to the memory controller in the time that it previously took to transmit  $W$  bits, by increasing the IO clock from its original frequency  $F$  to a new frequency  $F \times L$ .<sup>6</sup> For our two-layer example in Figure 6a, we can increase the IO frequency, and therefore the bandwidth, to twice that of the baseline 3D-stacked DRAM (Figure 5b) by having each layer send two sets of 64-bit data at  $2F$  frequency.

#### 4.1.1 Implementation Details

Figure 7 compares the detailed organizations of existing 3D-stacked DRAM (Figure 7a), where all layers share all TSVs, and a 3D-stacked DRAM using Dedicated-IO (Figure 7b). To clearly explain the access scenarios for both cases, we use a simplified organization of 3D-stacked DRAM, consisting of two layers, and we focus only on a two-bit subset of each layer (i.e., it contains two TSVs as IO for data transfer, and each layer has two bitlines that can deliver two bits every clock period, which is  $1/F$  for the baseline). In this example, the Dedicated-IO-based 3D-stacked DRAM dedicates the left TSV to the lower stack layer, and the right TSV to the upper layer.

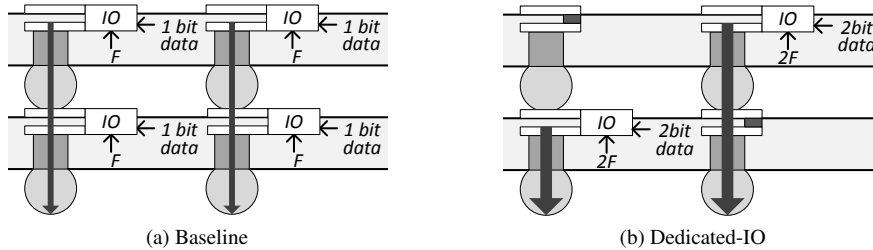


Figure 7: Simultaneous Multi Layer Access with Dedicated-IO

<sup>6</sup>Naturally, the magnitude of  $L$  is limited, as the frequency  $F \times L$  cannot exceed the total attainable frequency for the peripheral interface and IO interface.

In the case of the baseline 3D-stacked DRAM, only one selected layer (the upper layer in this example) can transfer data during a single clock period. Therefore, the upper layer transfers one bit per TSV (two bits in total). On the other hand, for the Dedicated-IO-based 3D-stacked DRAM, both layers are transferring data through the dedicated TSV of each layer. Note that it is still possible to increase the IO frequency of 3D-stacked DRAM, as conventional DRAMs have a much higher IO frequency (see Section 2). Therefore, the Dedicated-IO-based 3D-stacked DRAM can transfer two bits of data *from each layer* during one baseline clock period ( $1/F$ ) by doubling the IO frequency. As a result, Dedicated-IO enables twice the bandwidth of the conventional 3D-stacked DRAM by (i) simultaneously accessing both layers (doubling the available data per TSV) and (ii) transferring data at doubled frequency.

#### 4.1.2 Implementation Overhead

While Dedicated-IO enables much higher bandwidth, there are several drawbacks in this implementation on a real memory system. First, each layer of the 3D-stacked DRAM has physically different connections. Therefore, the manufacturing cost of the Dedicated-IO-based 3D-stacked DRAM may be higher than conventional 3D-stacked DRAM. Second, with an increasing number of layers, the IO clock frequency scales linearly, resulting in higher dynamic energy consumption.

### 4.2 Cascaded-IO

Cascaded-IO reorganizes the IO interface in order to address the design shortcomings of Dedicated-IO, through the use of time-sliced multiplexing. As Figure 6b shows, the IO interface of each layer has a multiplexer that can transfer either  $W$  bits of its own data, or  $W$  bits of data from an upper layer. By performing these transfers at  $F \times L$  frequency, each layer can send  $W$  data over the baseline clock period  $1/F$  without having to partition the data (i.e., each layer gets to transmit its data every  $L$  clock cycles). This mechanism is only possible in a 3D-stacked DRAM that contains two independent ports connected by peripheral circuits, as we discussed in Section 3.

#### 4.2.1 Implementation Details

Figure 8a shows the vertical structure of Cascaded-IO on a four-layer 3D-stacked DRAM. We again turn to a simplified example to illustrate the operation sequence, this time focusing on a one-bit subset of each layer, and with all bits connected to the same TSV. In this setup, the clock frequency, with respect to the baseline DRAM clock frequency  $F$ , will be  $4F$ . The data interface in each layer contains a multiplexer that selects from one of two paths: (i) a connection to the upper stack layers to transmit its bit of data, and (ii) a data wire from the layer's global sense amplifiers, connected to its own cell.

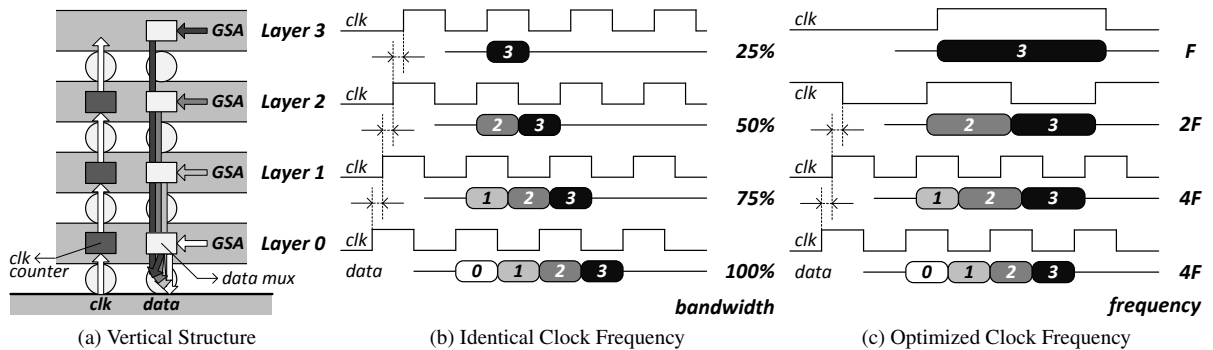


Figure 8: Simultaneous Multi Layer Access with Cascaded-IO

In a Cascaded-IO-based memory system, each layer will first fetch its own data from its cell. The multiplexer will then drive this data as the first data transfer, which will take the data from a layer and send it to the next layer below. Over the next three cycles, the layer will send data that it received from the layer above it. In essence, this will pipeline the data of all four layers over the four cycles, as shown in Figure 8b. Layer 1, for example, will drive its own data in the first cycle, sending it to Layer 0. In the second cycle, Layer 1 will receive Layer 2's data, and will now send that down to Layer 0. During this second cycle, Layer 2 will receive the data from Layer 3, sending that down to Layer 1 so that in cycle three, Layer 1 can now send the Layer 3 data down to Layer 0.



One side effect of this pipelined approach is that not all layers contain useful data to send to its lower level every cycle. In our example, Layer 1 has nothing to send in the fourth cycle, and Layer 3, with no layers above it, only sent useful data in the first of the four cycles. As we show in Figure 8b, the layer-by-layer TSV bandwidth utilization grows from 25% at Layer 3 up to 100% at Layer 0. Since the bandwidth in these upper layers is going to be unused, it is wasteful to still clock them at  $4F$  frequency. We exploit this tiered utilization to lower the energy utilization of higher layers, by simply running these layers at lower frequencies. We show the detailed effects of this energy reduction in Section 6.

While phase-locked loops (PLLs [25]) are a conventional way to generate clocks of arbitrary frequency, they consume large amounts of energy, making them an infeasible option for our heterogeneous clock frequency domains. Instead, Cascaded-IO adopts simple *two-bit clock counters*, which, when enabled, can generate a half-frequency clock. Figure 8a also shows the organization of the clock path. The clock signal typically originates from the memory controller. This clock signal is connected to the bottom layer of the 3D-stacked DRAM, and propagates toward the top layer. On the path to this clock propagation, Cascaded-IO inserts one clock counter per layer, which, when activated, can half the clock frequency.

Since these clock counters only perform simple divide-by-two, our clock path can only generate frequency values split by powers of two. Therefore, the ideal frequency for Layer 1,  $3F$ , cannot be generated. Instead, we must continue to use a  $4F$  frequency clock. Layer 2, however, can be driven at only  $2F$ , while Layer 3 can be driven at frequency  $F$ . In general, the lower half of the  $L$  layers will always run at frequency  $F \times L$ , while a quarter of them will run at frequency  $\frac{F \times L}{2}$ , an eighth of them at frequency  $\frac{F \times L}{4}$ , etc. The uppermost layer will always run at frequency  $F$ . Figure 8c illustrates how the timings and data transfer work with our reduced clock mechanism.<sup>7</sup>

#### 4.2.2 Implementation Overhead

To implement Cascaded-IO-based IO interfaces, each DRAM chip needs to include three components: (i) one clock counter, and (ii) a data multiplexer for every TSV, and (iii) control logic to switch the multiplexers between data sources (either connecting to the upper layer, or driving the contents of its own cells) based on the clock. The counter and multiplexer have very simple design, and only require a small number of transistors, resulting in negligible area overhead. For example, in Layer 0 from Figure 8, the multiplexer control consists of a two-bit counter, which connects its IO drivers to the lower TSVs when the counter value is  $2'b00$ . For all other values, the control will switch the multiplexer to connect the input from the upper layer to the lower TSV. Since only the counter is necessary, the overhead and complexity of the control logic are also negligible.

#### 4.2.3 Power Consumption Overhead

Due to the different frequencies and data bandwidths at each layer, the bottom layer in Cascaded-IO will indeed consume greater power. As the power network is driven from the bottom, power delivery will be strongest in the lower layer of 3D-stacked DRAM, and significantly weaker in the upper layers [31]. In these upper layers, Cascaded-IO provides a new solution to reduce power consumption, which as a byproduct allows the layers of Cascaded-IO to align well with the power network strength at each layer. While the bottom layer operates at  $4X$  frequency, the upper layers reduce their clock frequencies to match their much lower bandwidth requirements (e.g., the topmost layer is clocked at  $1X$  frequency). Therefore, while the physical design of each layer remains homogeneous, we can minimize the increase in power by introducing heterogeneity in the layers' operating frequencies. Such an approach has much lower power consumption than if we were to run all of the layers at  $4X$  frequency (as is done in Dedicated-IO).

With its strong power delivery, the bottom layer can deliver similar per-pin bandwidth as other high performance DRAMs (e.g., GDDR5, which has maximum  $6Gbps/pin$ ). It is important to note that a group of IO pins (usually 2 – 4 IO pins per group) has a dedicated power source (both  $V_{ddq}$  and  $V_{ssq}$ ). Therefore, it is reasonable to estimate the IO power capacity based on the IO clock frequency (or per-pin bandwidth) rather than overall per-chip bandwidth. Considering that, the bottom layer should operate as high as  $6Gbps/pin$  without power issues (as GDDR5 does), which is much higher than the highest IO frequency ( $800Mbps/pin$ ), which we assume in our evaluations. We discuss detailed power analysis in Section 6.

<sup>7</sup>To avoid issues due to clock skew, the data multiplexer for a layer is actually only responsible for synchronizing its own data. Afterwards, when the multiplexer switches to “receive” data from the upper layer, it simply connects a bypass path from the upper layer to the lower layer for the remaining cycles. This allows the upper layer to directly send its data to the bottommost layer. Figure 8c depicts the timing of this cut-through mechanism with skew. For example, in Layer 1, the data switches from Layer 2's bit to Layer 3's bit in the middle of the Layer 1 clock cycle, since the multiplexer in Layer 1 is now directly connecting Layer 2 to Layer 0 without synchronization.

## 5 Rank Organizations with Multiple Layers

In DRAM, a rank is a collection of DRAM chips that operate in lockstep. A rank-based organization allows pieces of a row to be split up across these chips, since they logically appear to be a single structure. Existing 3D-stacked DRAM treats each layer as a single rank, as only one of these layers can be accessed at a time. SMLA opens up new possibilities for the rank organization, since multiple DRAM layers can now be accessed simultaneously. In this section, we explore two such organizations: *Multi-Layer Ranks* (MLR) and *Single-Layer Ranks* (SLR), which represent a key trade-off between data transfer latency and rank-level parallelism.

Figure 9 shows the timeline of serving requests using MLR and SLR in a two-layer stacked DRAM, with either Dedicated-IO or Cascaded-IO. To show the data flow of Dedicated-IO, the IO bus is divided into two groups (*IO Group 0* and *IO Group 1*) that are each connected to only one of the two stack layers in Dedicated-IO (To compare Cascaded-IO and Dedicated-IO, we keep using two IO groups in the timeline for Cascaded-IO, even though Cascaded-IO does not need to divide IO bus into IO groups.) MLR merges both layers into a single rank, while SLR provides two ranks. In both cases in our example, a rank consists of two banks, to avoid rearchitecting the chips. (In the case of MLR, this will result in banks with twice the capacity.) We illustrate the timeline for serving three requests, each of which requires four data transfers through an IO group (two data transfers through the entire IO).

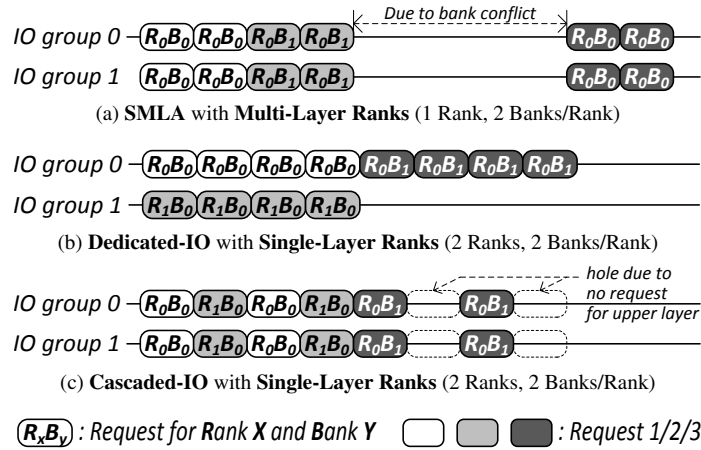


Figure 9: Request Service Timeline for Two Rank Designs

**SMLA with Multi-Layer Ranks.** MLR is similar to the current rank organization of non-stacked DRAM, with multiple layers sharing a common command interface. In our example MLR-based memory system with either Dedicated-IO or Cascaded-IO, the data is transferred through both IO groups, requiring *two* clock cycles to serve a request (*Request 1* in Figure 9a). Note that while both Dedicated-IO and Cascaded-IO have the same timeline when observing IO groups, the data served resides in two different physical locations. In Dedicated-IO, the data for *IO Group 0* comes from one layer (e.g., the bottom layer) and the *IO Group 1* data comes from the other (top) layer. In Cascaded-IO, the first piece of data for both IO groups come from the bottom layer, while the second piece comes from the top layer. The third request in our sequence has an access to the same bank and rank as the first request. Therefore, there will be a delay before serving the third request due to a bank conflict [17].

**SMLA with Single-Layer Ranks.** In a Dedicated-IO-based memory system with SLR (Figure 9b), each layer is a rank, and has its own IO group. Therefore, a request to a rank will only transfer data over its assigned group, requiring four clock cycles to serve a request. On the other hand, Cascaded-IO partitions its data transfer cycles in a round robin fashion, with each layer having an assigned time slot (one every  $L$  cycles). In Figure 9c, *Rank 0* corresponds to the bottom layer, and *Rank 1* to the top layer. We assign the lower layer to the first of the two time slots. Therefore, the odd data bursts correspond to *Rank 0* requests, while the even data bursts correspond to *Rank 1* requests. In this way, the first and third data bursts represent the first complete request to *Rank 0*. When our third request is serviced, since the Cascaded-IO slot assignments are fixed, there will be a hole in between its two data bursts, since *Rank 1* will not be serving anything.

**Trade-Off: MLR vs. SLR.** For our two-layer memory system, MLR can fully service an individual request within two cycles, through the use of both IO groups. In contrast, SLR takes up to four cycles to service each request, but can deliver *two* requests in that time, since it can overlap latencies for accesses to different ranks. MLR therefore better supports latency-bound applications that do not issue large numbers of memory requests. SLR, on the other hand, is

better tuned for memory-level parallelism, as it exposes a greater number of ranks. As we can see in Figure 9, the third request experiences fewer delays in SLR because the larger number of available ranks reduces the probability of conflicts.

While we have considered two extremes of rank organization (one layer per rank vs. using all layers in a rank), it is possible to pick a design point in the middle (e.g., two ranks in a four-layer memory, with two layers per rank). We leave the evaluation of such organizations as future work.

## 6 Energy Consumption Analysis

In order to deliver greater memory bandwidth, SMLA increases the memory channel frequency, which can change energy consumption significantly. To analyze the energy efficiency of our mechanisms, we: (i) observe the trend of DRAM energy consumption at different operating frequencies, (ii) determine which components of energy consumption are coupled with clock frequency, and (iii) estimate the overall energy consumption of our proposed mechanisms.

Figure 10 plots the current specifications of DDR3 at four different IO frequencies (1066, 1333, 1600, and 1866MHz, based on manufacturer specifications [22]). Each current represents a specific property of DRAM — either periodically-performed operations (IDD1) or current due to staying in a particular state (IDD2N/P and IDD3N/P).

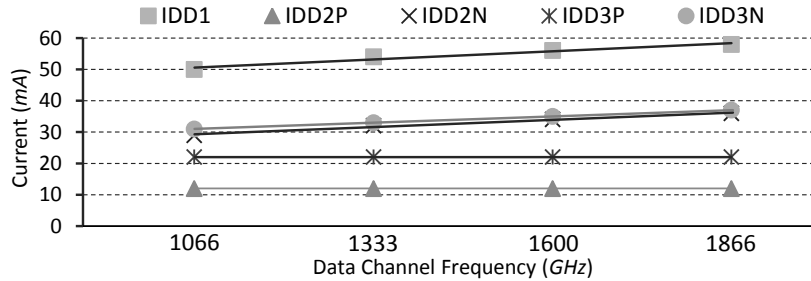


Figure 10: DDR3 Current Trend over Channel Frequency

In power-down mode (IDD2P and IDD3P), DRAM exploits current reduction techniques that stop the internal clock, resulting in no variation at different frequencies. Standby currents at different frequencies (IDD2N and IDD3N) show large variation (about a 20 – 30% energy reduction when cutting the frequency in half). Therefore, DRAM consumes a significant fraction of overall energy to maintain the clock propagation network. When accessing DRAM cells, the absolute amount of the current reduction from halving the frequency is similar to the reduction in standby mode. However, due to the high energy consumption when accessing cells (either activate/precharge or read/write), the amount of reduced energy is relatively small (about 10 – 15%) compared to the reduction in standby modes. We draw two observations from these results. First, increasing the clock frequency leads to greater current flow (larger energy consumption), with the relationship between energy consumption and clock frequency mostly linear. Second, the slope of the current-frequency trend is similar in normal mode for all frequencies.

Using this observation along with DRAM energy estimation tools [27, 37, 23], we extract two current sources: (i) current coupled with the clock, and (ii) current decoupled from the clock. We then estimate the current consumptions at different frequencies. Table 1 presents energy consumption for three different operating conditions (power-down, standby, and access) at four different operating frequencies (200, 400, 800, and 1600MHz), estimated from the measured energy consumption of 3D-stacked DRAM [3]. We see that a large fraction of DRAM energy consumption in standby modes is coupled with the operating clock frequency.

Clock Frequency (MHz)	200	400	800	1600
Power-Down Current (mA)	0.24	0.24	0.24	0.24
Precharge-Standby Current (mA)	4.24	5.39	6.54	8.84
Active-Standby Current (mA)	7.33	8.50	9.67	12.0
Active-Precharge without Standby (nJ)	1.36	1.37	1.38	1.41
Read without Standby (nJ)	1.93	1.93	1.93	1.93
Write without Standby (nJ)	1.33	1.33	1.33	1.33

Table 1: Energy Consumption Estimation

## 7 Methodology

We use a cycle-accurate in-house simulator that models 3D-stacked memory and whose front-end is based on Pin [21, 26]. We use existing 3D-stacked DRAM, WideIO, as our baseline. More recent 3D-stacked DRAM proposals (WideIO2, HMC, HBM) can enable more bandwidth at the expense of area and power consumption (as shown in Section 2.3). As we discussed, the key advantage to our mechanisms (Dedicated-IO and Cascaded-IO) is our ability to deliver competitive bandwidth to these high-bandwidth proposals at a much lower global sense amplifier cost. It is also important to note, however, that simply performing bandwidth scaling (as WideIO2/HMC/HBM have done) will eventually be constrained by the area and power overhead of these extra global sense amplifiers. We show that our mechanisms overcome this challenge with a large reduction in sense amplifier count, and can still be used on top of these more recently-proposed DRAM architectures to further increase off-chip bandwidth.

We evaluate our mechanisms for 3D-stacked DRAM with two/four/eight layers, with a corresponding increase in IO frequency of 2/4/8 times, respectively, over conventional 3D-stacked DRAM. We evaluate two rank organizations (Single-Layer Rank and Multi-Layer Rank). Table 2 summarizes the parameters for the baseline 3D-stacked memory system and our proposed memory systems, which consist of four stacked layers and operate at 4X of the baseline system’s IO frequency.

IO interface	Baseline	Dedicated-IO		Cascaded-IO	
Rank Organization	SLR	MLR	SLR	MLR	SLR
Number of Ranks	4	1	4	1	4
Clock Freq. (MHz)	200	800	800	800	800
Bandwidth (GBps)	3.2	12.8	12.8	12.8	12.8
Data Trans. Time (ns)	20.0	5.0	20.0	5.0	18.1†
Sim. Multi Layer Acc.	1	4	4	4	4

★ Global Param.: 4 Layers, 2 Banks/Rank, 128 IOs/Chan., 64 Byte/Req.

† Avg. Data Transfer Time of Layers (Bottom : 16.25/17.5/18.75/Top : 20)

Table 2: 3D-Stacked DRAM Configurations Evaluated

Table 3 shows the evaluated system configuration. We use a one channel memory system for our single-core evaluations, and a four channel system for multi-core evaluations (identical to the configuration used in Wide-IO [9]).

Component	Parameters
Processor	1 – 16 cores, 3.2GHz, 3-wide issue, 8 MSHRs, 128-entry instruction window
Last-Level Cache	64B cache-line, 16-way associative, 512KB private cache-slice per core
Memory Controller	64/64-entry read/write queues/controller, FR-FCFS scheduler [29]
Memory System	2 – 8 layer stacked DRAM, 1 – 4 channel

Table 3: Configuration of Simulated System

We use 31 applications from the SPEC CPU2006, TPC [36], and STREAM [34] applications suites. For single-core studies, we report results that are averaged across all applications. For our multi-core evaluation, we generate 16 multi-programmed workloads for each case (4–16 cores / 2–8 stacked layers) by randomly selecting from our workload pool.

We execute all applications for 100 million instructions, similar to many prior works [35, 24, 4, 17] that also studied memory systems. For multi-core evaluations, we ensure that even the slowest core executes 100 million instructions, while other cores continue to exert pressure on the memory subsystem. To measure performance, we use instruction throughput (IPC) for single-core systems and *weighted speedup* [33] for multi-core systems.

## 8 Evaluation

In our experiments for both single-core and multi-core systems, we compare three different 3D-stacked DRAM designs: (i) the baseline conventional 3D-stacked DRAM, (ii) *Dedicated-IO*, (iii) *Cascaded-IO* (the latter two are SMLA-based designs) both in terms of performance and energy efficiency.

### 8.1 Single-Core Results

Figure 11 compares SMLA-based designs (*Dedicated-IO* and *Cascaded-IO*) to the baseline with conventional 3D-stacked DRAM.<sup>8</sup> All three 3D-stacked DRAM designs have four stacked layers (Table 2). By accessing those four layers simultaneously, SMLA-based mechanisms provide four times more bandwidth compared to the baseline. Each mechanism was evaluated with the two rank organizations: Single-Layer Rank (SLR, 4 ranks per channel in Figure 11b), and Multi-Layer Rank (MLR, 1 rank per channel in Figure 11a). For each application, the figures show two metrics: (i) performance improvement of SMLA-based mechanisms compared to the baseline, and (ii) the power reduction of SMLA-based mechanisms relative to the baseline. We draw three conclusions from the figure.

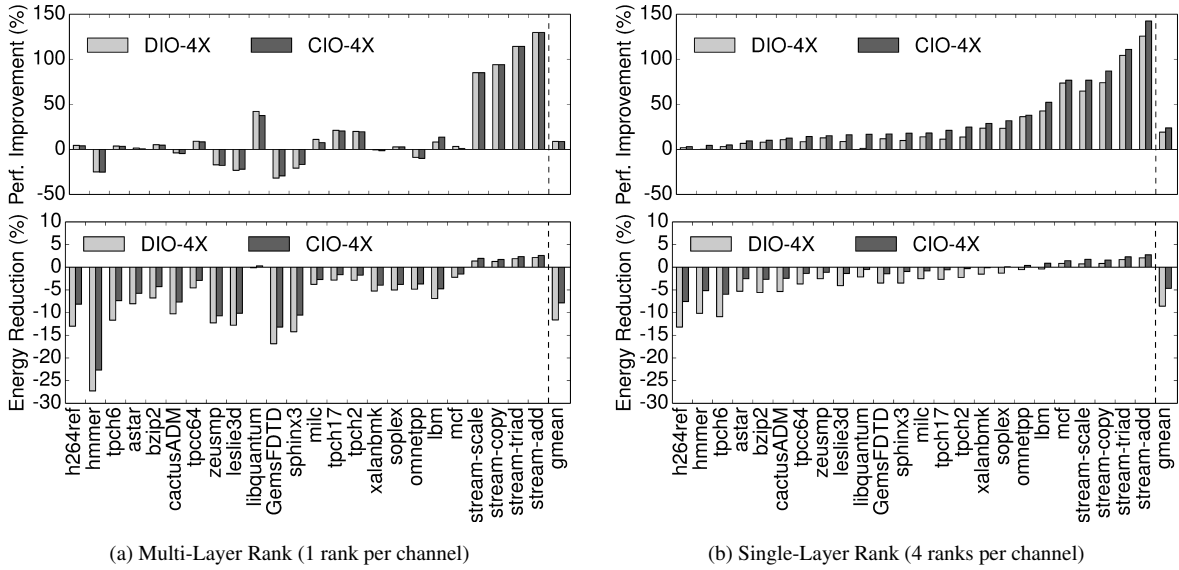


Figure 11: Performance and Energy Efficiency Evaluations on a Single-Core System

**Performance with Rank Organizations.** First, *SMLA-based mechanisms provides significant performance improvement on average in both rank organizations. However, individual application shows different performance benefit based on the rank organizations.*

In SLR (Figure 11b), for most applications, both of our mechanisms improve performance significantly compared to the baseline. On average, *Dedicated-IO* and *Cascaded-IO* improve performance by 19.2% and 23.9%, respectively, compared to the baseline. Intuitively, performance benefits are higher for applications with higher memory intensity (MPKI of the seven left-most applications is less than 8, while the average MPKI of the seven right-most applications is more than 35.).

In MLR (Figure 11a), while SMLA-based mechanisms improve performance for most applications compared to the baseline (by 8.8% on average for both mechanisms), there are a few cases performance degrades. Note that even though both *Dedicated-IO* and *Cascaded-IO* enable much higher bandwidth and lower latency, these mechanisms with MLR have only one rank per channel. This reduces the number of opportunities for rank-level parallelism. As a result, several applications (e.g., *hmmer*, *zeusmp*, *leslie3d*, *GemsFDTD*, *sphinx3*, *omnetpp*) will benefit more from having more rank-level parallelism (in the baseline) than from having higher bandwidth and lower latency (in MLR).

<sup>8</sup>We present 24 applications whose *MPKI* (last-level cache Misses-Per-Kilo-Instructions) are larger than 1. The other benchmarks (*MPKI* < 1) show performance changes of less than 1%. However, the provided average (geometric mean) results for both performance and energy include all 31 workloads.

For most applications, SMLA-based mechanisms with SLR provide better performance improvement compared to the corresponding benefits with MLR. There are, however, a few noticeable exceptions (e.g., `libquantum`, `h264ref`) – applications where Dedicated-IO and Cascaded-IO show better performance improvement with MLR. These applications are more latency sensitive, and hence they benefit more from low-latency in MLR than from rank-level parallelism in SLR.

**Performance with SMLA Designs.** Second, *Cascaded-IO usually provides better performance improvement than Dedicated-IO in SLR*. This is because Cascaded-IO has lower average latency than Dedicated-IO in SLR. However, in MLR, both mechanisms have same latency (shown in Table 2). Furthermore, upper layers in Cascaded-IO operate at lower frequency, leading to reduction in the command and address bandwidth for the upper layers. As a result, in MLR, Dedicated-IO shows a slightly better performance than Cascaded-IO.

**Energy Consumption.** Third, *Cascaded-IO shows better energy efficiency compared to Dedicated-IO due to reduction in the frequency of the third and fourth layers*. However, due to the increase in the overall frequency, on average (and especially for lower layers), both Dedicated-IO and Cascaded-IO consume more energy compared to the baseline (8.6% and 4.6% more with SLR and 11.6% and 7.8% with MLR, respectively).

## 8.2 Multi-Core Results

Figure 12a shows the system performance improvement for our mechanisms in multi-programmed workloads. Similar to single-core results with SLR, both Dedicated-IO and Cascaded-IO provide significant performance improvement in all system configurations. On average, Dedicated-IO obtains a 14.4%/28.3%/50.4% weighted speedup for 4/8/16-core systems, respectively, while Cascaded-IO provides a further performance bump of 18.2%/32.9%/55.8%. However, due to reduction in rank-level parallelism, the performance benefits of our mechanisms using MLR are lower than with SLR.

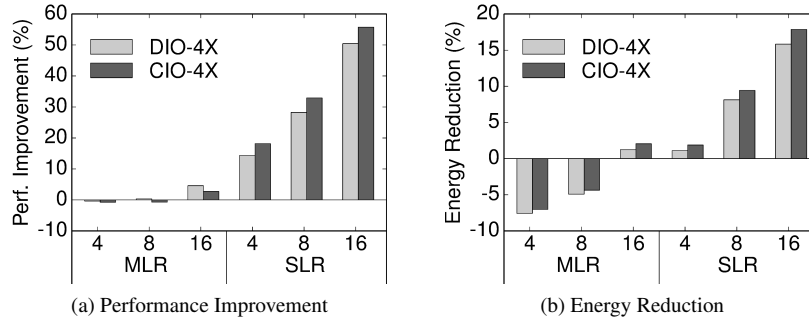


Figure 12: Multi-Core Evaluation (4/8/16 Cores, 4 Layers)

Our mechanisms increase power consumption (energy/time) due to increased IO frequency and enabling more IO bandwidth. However, our mechanisms can avoid overall memory system energy increase, mainly due to reduced execution time in our applications. We see that unlike in the single-core case, our mechanisms with SLR can actually deliver significant energy reductions as the number of cores (and therefore the memory contention) increases. In such cases, where there is a higher bandwidth demand, SMLA alleviates much of this contention, resulting in shorter overall execution times. Figure 12b shows these savings with SLR, which on average are 1.9%/9.4%/17.9% for 4/8/16 cores, respectively. Unlike SLR, MLR is geared towards latency and not parallelism, and as such is unable to ease the contention pressure as well, resulting in increased energy consumption.

## 8.3 Sensitivity to the Number of Stacked Layers

The maximum bandwidth improvement in our mechanisms depends on the number of stacked-layers whose global bitline interfaces can operate simultaneously. For example, in a two layer-stacked memory system, our mechanism can enable twice the bandwidth of the conventional 3D-stacked DRAM (4/8 times the bandwidth for 4/8 layer-stacked DRAM, respectively). Figure 13 shows the system performance improvement and energy reduction of our mechanisms in 2 – 8 layer-stacked DRAM. We use eight-core multi-programmed workloads and our evaluated memory system has four channels.

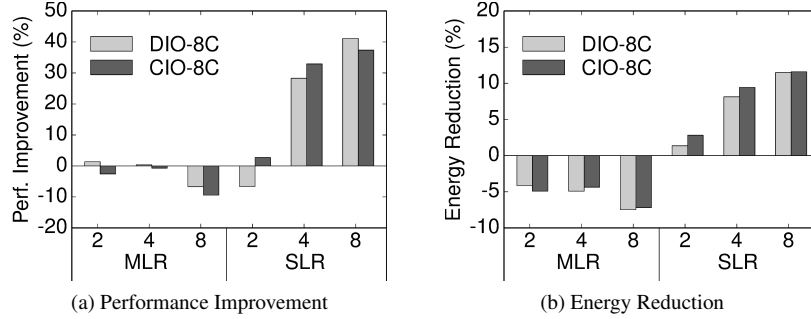


Figure 13: Sensitivity for Layer-Count (8 Cores, 2/4/8 Layers)

Two observations are in order. First, as expected, our mechanisms with SLR provide better performance and energy efficiency that grows with the number of stacked layers in DRAM. At the same time, we observe less performance benefits in eight layer-stacked DRAM with MLR, mainly due to reduction in the number of ranks (one rank in MLR-based system vs. eight ranks in the baseline system). Second, a 3D-stacked DRAM with more layers, (especially, eight layer-stacked DRAM), has better performance with Dedicated-IO than with Cascaded-IO due to reduction in frequency of the higher layers in Cascaded-IO. This, in turn, leads to reduction in command bandwidth of upper layers.

#### 8.4 Energy Consumption with Varying Memory Intensity

Figure 14 shows the energy consumption of SMLA-based mechanisms, compared to the baseline, with varying memory intensity (MPKI) of micro-benchmarks in two different ways. First, we observe the absolute energy consumption, which is normalized to the baseline with 0.1 MPKI (Figure 14a). As expected, the energy consumption of all three mechanisms is growing with increase in memory intensity (21X more energy consumed at 51.2 MPKI than at 0.1 MPKI). Second, we analyze the relative energy increase when executing the same applications (Figure 14b). We observe that the amount of energy increase (relative to the baseline) significantly reduces at higher MPKIs. This is because DRAM consumes more energy to serve more memory requests, and the amount of energy consumed by these operations is not dependent on IO frequency. In other words, the absolute value of additionally consumed energy is fairly constant across all operating conditions, and relatively small compared to the amount of energy to serve memory requests.

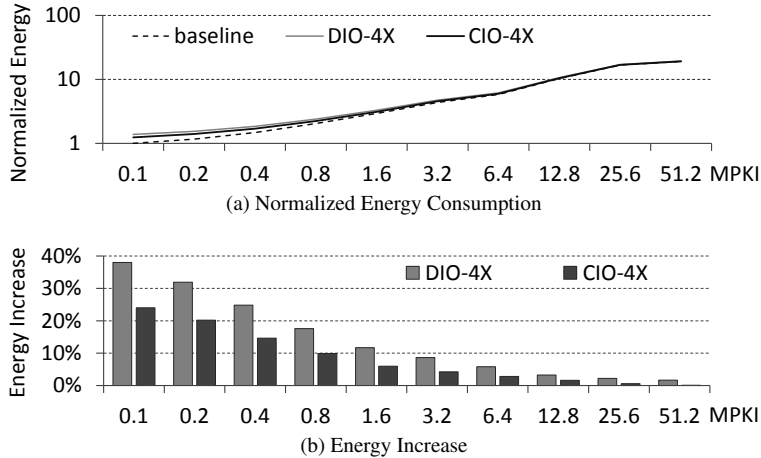


Figure 14: Memory Intensity vs. Energy Consumption

In all the cases, Cascaded-IO provides better energy efficiency than Dedicated-IO (about 30% lower energy overhead due to lower frequency of certain layers). Based on this analysis, we conclude that (i) SMLA-based designs can increase energy consumption, but this overhead is relatively small to the overall DRAM energy when running memory intensive applications, and it is also small compared to the overall system energy when applications are not memory intensive, and (ii) Cascaded-IO provides better energy efficiency than Dedicated-IO.

## 9 Related Work

To our knowledge, this is the first work that (i) enables higher bandwidth at low cost by leveraging the existing global bitline interfaces in multiple layers of 3D-stacked DRAM, and (ii) provides higher bandwidth with low energy consumption by optimizing IO frequency individually for each layer. In this section, we discuss the prior works that aim to improve memory system bandwidth.

**Bank-Group.** LPDDR3 [11] and DDR4 [10] categorize banks into multiple groups (bank-group), where each group owns a set of global sense amplifiers. LPDDR3 and DDR4 increase DRAM internal bandwidth by simultaneously accessing each bank-group, where our design provides higher bandwidth in 3D-stacked DRAM by aggregating the DRAM internal bandwidth in multiple layers. Potentially, these two approaches are orthogonal and can be applied together to further increase the bandwidth in 3D-stacked DRAM.

**High Bandwidth Memory.** HBM [12, 18] enables high bandwidth (128GBps) 3D-stacked DRAMs by (i) increasing DRAM internal bandwidth by adding extra global bitlines per chip (2X compared to Wide-IO), (ii) allowing simultaneous accesses to different bank-groups having their own set of global sense amplifiers (same as [11, 10]), and (iii) aggregating the bandwidth of each layer by assigning exclusive channels to each layer. On the other hand, we provide higher bandwidth without adding any extra bitline interface, by aggregating data available at different layers and transferring them at a higher frequency. HBM can achieve performance and energy efficiency similar to Dedicated-IO, but lower than Cascaded-IO, while integrating 2X global sense amplifiers than our mechanisms (as shown in Section 2.3).

**3D-Stacked DRAM Studies.** Prior works on 3D-stacked DRAM focused on utilizing the higher capacity of it as cache, and main memory [1, 19, 39]. A prior study focused on the architectural trade-offs in rank and channel organizations [20]. However, none of these works focus on solving the limited internal bandwidth of DRAM in presence of wider IO.

**Reconfiguring DIMM Organization.** Mini-rank [40] enables independent accesses to sub-divided ranks, similar to our mechanisms. However, our mechanisms enable more bandwidth by mitigating internal bandwidth bottleneck of 3D-stacked DRAM, which is not possible in Mini-rank. By integrating a buffer on DIMM, Decoupled-DIMM [41] enables higher frequency memory channel by decoupling memory channel from DRAM chips, similar to our mechanisms. However, our mechanisms can be implemented with small changes in 3D-stacked DRAM, while Decoupled-DIMM requires the expensive high performance buffer.

**Multi-Layer Rank Organization.** Loh [20] introduced a 3D-stacked DRAM that form a rank over multiple layers. That work assumes that all upper layers only have DRAM cells and that the bottom layer contains special logic to control all of the upper layers, which is not present in existing 3D-stacked DRAMs. However, Our mechanisms enable MLR within existing 3D-stacked DRAM architectures (where all layers have control logic and communicate only across IO) by enabling simultaneous multi-layer access, through minimal, uniform additions to the control logic.

## 10 Conclusion

In this work, we introduce Simultaneous Multi Layer Access (SMLA), a new IO organization for 3D-stacked DRAM that enables greater off-chip memory bandwidth at low cost. We identify that the major bandwidth bottleneck of existing 3D-stacked DRAM is the global bitline interface. We offer a cheaper alternative to adding more global bitlines, by instead exploiting the otherwise-idle internal bandwidth of multiple layers to drive a faster IO interface.

We study two IO implementations for SMLA, including Cascaded-IO, which time-multiplexes the IO bus across all of the DRAM layers and exploits its dataflow to reduce the energy impact of a faster-clocked memory. We evaluate SMLA over a wide array of applications, and show that our proposed mechanisms significantly improve performance and reduce energy consumption (on average 55%/18%, respectively, for 16-core multi-programmed workloads) over conventional 3D-stacked DRAMs. We conclude that SMLA provides a high performance and energy efficient IO interface for building modern (and future) 3D-stacked memory systems at low cost.

## Acknowledgments

We thank the SAFARI group members for the feedback and stimulating research environment they provide. We thank Uksong Kang, Jung-Bae Lee, and Joo Sun Choi from Samsung for their helpful comments. We acknowledge the



support of our industrial partners: Facebook, Google, IBM, Intel, Microsoft, Qualcomm, VMware, and Samsung. This research was partially supported by NSF (grants 0953246, 1212962, 1065112, 1320531), Semiconductor Research Corporation, and the Intel Science and Technology Center for Cloud Computing. Donghyuk Lee is supported in part by a Ph.D. scholarship from Samsung and the John and Claire Bertucci Graduate Fellowship. Gennady Pekhimenko is supported in part by a Microsoft Research Fellowship.

## References

- [1] B. Black, M. Annavaram, N. Brekelbaum, J. DeVale, L. Jiang, G. H. Loh, D. McCaule, P. Morrow, D. W. Nelson, D. Pantuso, P. Reed, J. Rupley, S. Shankar, J. Shen, and C. Webb. Die Stacking (3D) Microarchitecture. In *MICRO*, 2006. 16
- [2] D. Burger, J. R. Goodman, and A. Kägi. Memory bandwidth limitations of future microprocessors. In *SIGARCH Comput. Archit. News*, 1996. 1
- [3] K. Chandrasekar, C. Weis, B. Akesson, N. Wehn, and K. Goossens. System and circuit level power modeling of energy-efficient 3D-stacked wide I/O DRAMs. In *DATE*, 2013. 11
- [4] Y. Chou, B. Fahs, and S. Abraham. Microarchitecture Optimizations for Exploiting Memory-Level Parallelism. In *ISCA*, 2004. 12
- [5] J. Dean and L. A. Barroso. The Tail at Scale. In *Commun. ACM*, 2013. 1
- [6] Q. Harvard and R. Baker. A scalable I/O architecture for wide I/O DRAM. In *MWSCAS*, 2011. 4
- [7] C. Huyghebaert, J. Van Olmen, O. Chukwudi, J. Coenen, A. Jourdain, M. Van Cauwenberghe, R. Agarwahl, A. Phommahaxay, M. Stucchi, and P. Soussan. Enabling 10um pitch hybrid Cu-Cu IC stacking with Through Silicon Vias. In *ECTC*, 2010. 4
- [8] Hybrid Memory Cube Consortium. Hybrid Memory Cube. [www.hybridmemorycube.org](http://www.hybridmemorycube.org). 1
- [9] JEDEC. *JESD229 Wide I/O Single Data Rate (WideIO SDR)*, Dec. 2011. 5, 12
- [10] JEDEC. *JESD79-4 DDR4 SDRAM*, Sept. 2012. 16
- [11] JEDEC. *JESD209-3B Low Power Double Data Rate 3 (LPDDR3)*, Aug. 2013. 16
- [12] JEDEC. *JESD235 High Bandwidth Memory (HBM) DRAM*, Oct. 2013. 1, 2, 5, 16
- [13] JEDEC. *JESD229-2 Wide I/O 2 (WideIO2)*, 2014. 1, 2, 5
- [14] U. Kang, H.-J. Chung, S. Heo, S.-H. Ahn, H. Lee, S.-H. Cha, J. Ahn, D. Kwon, J.-H. Kim, J.-W. Lee, H.-S. Joo, W.-S. Kim, H.-K. Kim, E.-M. Lee, S.-R. Kim, K.-H. Ma, D.-H. Jang, N.-S. Kim, M.-S. Choi, S.-J. Oh, J.-B. Lee, T.-K. Jung, J.-H. Yoo, and C. Kim. 8Gb 3D DDR3 DRAM using through-silicon-via technology. In *ISSCC*, 2009. 1
- [15] B. Keeth, R. J. Baker, B. Johnson, and F. Lin. *DRAM Circuit Design. Fundamental and High-Speed Topics*. Wiley-IEEE Press, 2007. 2
- [16] J.-S. Kim, C. S. Oh, H. Lee, D. Lee, H.-R. Hwang, S. Hwang, B. Na, J. Moon, J.-G. Kim, H. Park, J.-W. Ryu, K. Park, S.-K. Kang, S.-Y. Kim, H. Kim, J.-M. Bang, H. Cho, M. Jang, C. Han, J.-B. Lee, K. Kyung, J.-S. Choi, and Y.-H. Jun. A 1.2V 12.8GB/s 2Gb mobile Wide-I/O DRAM with 4x128 I/Os using TSV-based stacking. In *ISSCC*, 2011. 1, 4, 5
- [17] Y. Kim, V. Seshadri, D. Lee, J. Liu, and O. Mutlu. A Case for Exploiting Subarray-Level Parallelism (SALP) in DRAM. In *ISCA*, 2012. 10, 12
- [18] D. U. Lee, K. W. Kim, K. W. Kim, H. Kim, J. Y. Kim, Y. J. Park, J. H. Kim, D. S. Kim, H. B. Park, J. W. Shin, J. H. Cho, K. H. Kwon, M. J. Kim, J. Lee, K. W. Park, B. Chung, and S. Hong. 25.2 A 1.2V 8Gb 8-channel 128GB/s high-bandwidth memory (HBM) stacked DRAM with effective microbump I/O test methods using 29nm process and TSV. In *ISSCC*, 2014. 1, 5, 16
- [19] G. Loh. Extending the effectiveness of 3D-stacked DRAM caches with an adaptive multi-queue policy. In *Micro*, 2009. 16
- [20] G. H. Loh. 3D-Stacked Memory Architectures for Multi-core Processors. In *ISCA*, 2008. 16
- [21] C.-K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. J. Reddi, and K. Hazelwood. Pin: building customized program analysis tools with dynamic instrumentation. In *PLDI*, 2005. 12
- [22] Micron. *2Gb: x4, x8, x16 DDR3 SDRAM*, 2006. 4, 11
- [23] Micron. *DDR3 SDRAM System-Power Calculator*, 2010. 11
- [24] O. Mutlu and T. Moscibroda. Stall-Time Fair Memory Access Scheduling for Chip Multiprocessors. In *MICRO*, 2007. 12
- [25] I. Novof, J. Austin, R. Kelkar, D. Strayer, and S. Wyatt. Fully integrated CMOS phase-locked loop with 15 to 240 MHz locking range and 50 ps jitter. In *JSSC*, 1995. 9
- [26] H. Patil, R. Cohn, M. Charney, R. Kapoor, A. Sun, and A. Karunanidhi. Pinpointing representative portions of large Intel Itanium programs with dynamic instrumentation. In *MICRO*, 2004. 12
- [27] Rambus. *DRAM Power Model*, 2010. 2, 6, 11
- [28] B. Razavi. *Design of Analog CMOS Integrated Circuits*. McGraw-Hill, 2000. 2
- [29] S. Rixner, W. J. Dally, U. J. Kapasi, P. Mattson, and J. D. Owens. Memory access scheduling. In *ISCA*, 2000. 12
- [30] B. M. Rogers, A. Krishna, G. B. Bell, K. Vu, X. Jiang, and Y. Solihin. Scaling the Bandwidth Wall: Challenges in and Avenues for CMP Scaling. In *ISCA*, 2009. 1

- [31] M. Shevgoor, J.-S. Kim, N. Chatterjee, R. Balasubramonian, A. Davis, and A. N. Udipi. Quantifying the Relationship Between the Power Delivery Network and Architectural Policies in a 3D-stacked Memory Device. In *MICRO*, 2013. 9
- [32] K. Smith, A. Wang, and L. Fujino. Through the Looking Glass: Trend Tracking for ISSCC 2012. In *IEEE Solid-State Circuits Magazine*, 2012. 1, 4
- [33] A. Snavely and D. M. Tullsen. Symbiotic jobscheduling for a simultaneous multithreaded processor. In *ASPLOS*, 2000. 12
- [34] STREAM Benchmark. [www.streambench.org](http://www.streambench.org). 12
- [35] J. Stuecheli, D. Kaseridis, D. Daly, H. C. Hunter, and L. K. John. The virtual write queue: Coordinating DRAM and last-level cache policies. In *ISCA*, 2010. 12
- [36] Transaction Processing Performance Council. . [www.tpc.org](http://www.tpc.org). 12
- [37] T. Vogelsang. Understanding the Energy Consumption of Dynamic Random Access Memories. In *MICRO*, 2010. 2, 6, 11
- [38] J. West, Y. Choi, and C. Vartuli. Practical implications of via-middle Cu TSV-induced stress in a 28nm CMOS technology for Wide-IO logic-memory interconnect. In *VLSIT*, 2012. 4
- [39] D. H. Woo, N. H. Seong, D. Lewis, and H.-H. Lee. An optimized 3D-stacked memory architecture by exploiting excessive, high-density TSV bandwidth. In *HPCA*, 2010. 16
- [40] H. Zheng, J. Lin, Z. Zhang, E. Gorbatov, H. David, and Z. Zhu. Mini-rank: Adaptive DRAM architecture for improving memory power efficiency. In *MICRO*, 2008. 16
- [41] H. Zheng, J. Lin, Z. Zhang, and Z. Zhu. Decoupled DIMM: building high-bandwidth memory system using low-speed DRAM devices. In *ISCA*, 2009. 16