EFFICIENT DESIGN AND COMMUNICATION FOR 3D STACKED DYNAMIC MEMORY

An Undergraduate Research Scholars Thesis

by

ANDREW DOUGLASS

Submitted to the Undergraduate Research Scholars program at
Texas A&M University
in partial fulfillment of the requirements for the designation as an

UNDERGRADUATE RESEARCH SCHOLAR

Approved by Research Advisor: Dr. Sunil Khatri

May 2017

Major: Computer Engineering

TABLE OF CONTENTS

Page

ABSTRACT 1

ACKNOWLEDGMENTS 2

NOMENCLATURE 3

CHAPTER

I. INTRODUCTION 6

History of DRAM Architectures 6
Rambus DRAM 10
Graphics DRAM 11
Current DIMM Issues 12
3D Stacked Memory 15

II. RING BASED MEMORY 19

Skew Cancellation for HBM 19
Massed Refresh 20
Simultaneous Multi Layer Access 20
Ring-Based Interconnect 20

III. SIMULATION RESULTS 26

16nm Process 26
3D Clock Ring 27
Current DRAM Speeds 29
Low Power Ring Memory 36
High Performance Ring Memory 43

IV. CONCLUSION 49

Importance 49
Future Research 49

REFERENCES 51

APPENDIX: MEMORY SIMULATION DATA TABLES 54

ABSTRACT

Efficient Design and Communication for 3D Stacked Dynamic Memory

Andrew Douglass
Department of Electrical and Computer Engineering
Texas A&M University

Research Advisor: Dr. Sunil Khatri
Department of Electrical and Computer Engineering
Texas A&M University

As computer memory increases in size and processors continue to get faster, memory becomes an increasing bottleneck to system performance. To mitigate the slow DRAM memory chip speeds, a new generation of 3D stacked DRAM will allow lower power consumption and higher bandwidth. To communicate between these chips, this paper proposes the use of ring based standing wave oscillators for fast data transfer. With a fast clocking scheme, multiple channels can share the same bus to reduce TSVs and maintain similar memory latencies. Simulations with the new clocking scheme and data transfers are performed to show the improvements that can be made in memory communication. Experimental results show that a ring based clocking scheme can obtain two times the speed up of current stacked memory chips. Variations of this clocking scheme can also provide half the power consumption with comparable speeds. These ring-based architectures allow higher memory speeds without compromising the complexity of the hardware. This allows the ring-based memory architecture to trade off power, throughput, and latency to improve system performance for different applications.

ACKNOWLEDGEMENTS

I would like to thank my mentor, Dr. Sunil Khatri, for his guidance and continual support throughout the course of this research project. I would also like to thank Abbas Fairouz for his help with SPICE simulations and the time he dedicated to my questions.

Thanks also go to my friends, colleagues, and the faculty staff for making my time at Texas A&M University a great experience. I also want to extend my gratitude to the staff of the Undergraduate Research Scholars program, which provided many training sessions and assistance to all students involved in the program.

Finally, I want to thank my mother and father for being role models to me and always supporting my goals.

NOMENCLATURE

DRAM	Dynamic Random Access Memory (DRAM) is a form of volatile memory that involves a transistor and a capacitor to store bits of information.
JEDEC	The Joint Electron Device Engineering Council (JEDEC) is an organization made up from many industry leaders who create the standards in DRAM memory.
DIMM	Dual In-line Memory Module (DIMM) is a printed circuit board that contains multiple random access memory chips connected to a large bus with separate electrical contacts on each side of the module.
SO-DIMM	Small Outline Dual In-line Memory Module (SO-DIMM) is a smaller version of DIMM that is used for devices where space is limited (e.g. Laptops, Tablets, and Routers).
DMC	Dynamic Memory Controller (DMC) is the hardware device that translates addresses, queues requests, and communicates with memory modules.
MC	Memory Chips (MCs) are integrated circuits found on the DIMM printed circuit board that contain memory arrays to store data as bits.
Channel	An independent bus that communicates between the DMC and the DIMM, which often consists of a data portion, a control portion, and an address portion.

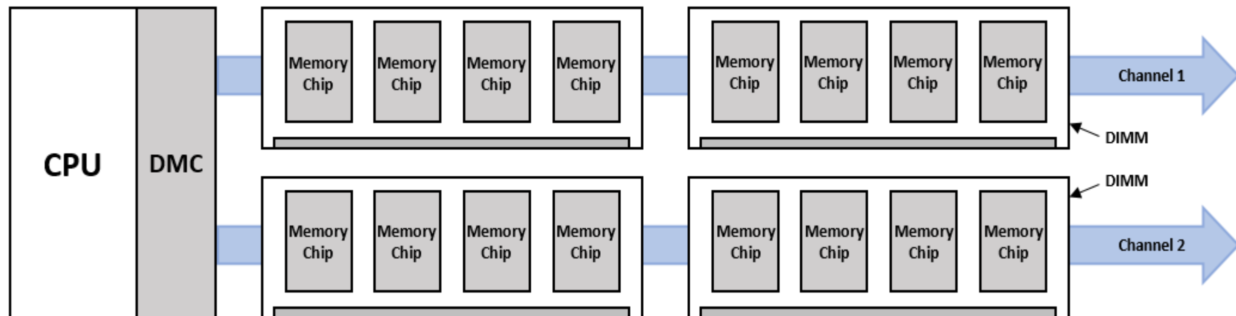

Figure 1: Multi-channel DRAM architecture.

Rank	A set of DRAM memory chips that share a control bus and operate simultaneously to provide the total bits for the data bus (shown in Figure 2).

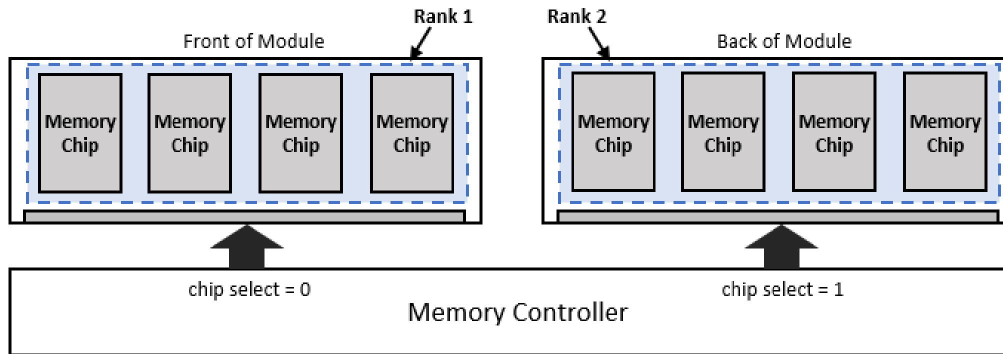

Figure 2: Both sides of a DIMM with ranks labeled.

Bank — A subset of memory arrays that operate independently to allow pipelining of requests to different banks.

Bit-line — Bit-lines represent columns in a memory array that fill with the value from a memory cell during reading and drive to the desired value when writing (shown in Figure 3).

Word-line — Word-lines represent rows in a memory array that drive to a high voltage when reading from or writing to the specified row (shown in Figure 3).

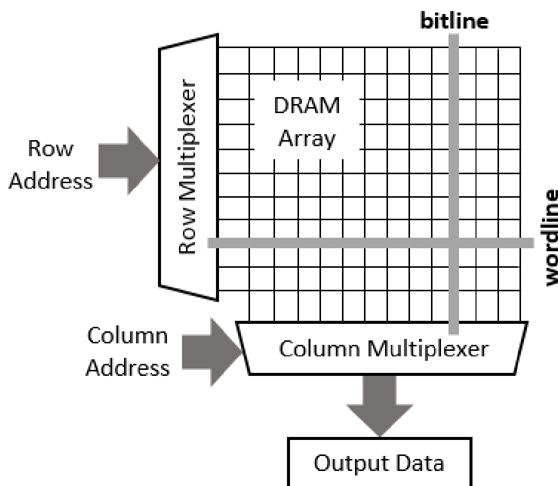

Figure 3: DRAM array bit-lines and word-lines.

DDR — Double Data Rate (DDR) is a computer bus that transfers data on both the positive edge and negative edge of the clock signal.

Sense Amps — A circuit used to detect small voltage changes on the bit-lines when reading data to determine what state (1 or 0) the specified capacitor was storing.

Refresh — The process of occasionally reading data into the sense amps and immediately writing that data back to the capacitors to prevent the loss of information.

Skew — Skew is the maximum time difference that it takes for a signal to arrive at one endpoint compared to another endpoint in the same distribution network.

H-Tree — Named for the shape it resembles, it is a clock distribution scheme used to minimize clock skew between its end points by maintaining similar delays through each signal path.

I/O Bus — Input/Output (I/O) bus is the set of pins on the DIMM that carry the data to the module during writes and from the module during reads.

PLL — Phase-locked loop (PLL) is a control system that outputs a signal whose phase is the same as the input reference signal.

SWO — Standing Wave Oscillator (SWO) is a type of oscillator circuit with an inverter pair on one end and a short on the other to create a clock signal with the same phase at every point along its resonant wires.

CHAPTER I

INTRODUCTION

Though cheaper and larger, forms of non-volatile memory have long latencies. This can severely slow down the time it takes for a system to execute a desired program. To overcome this issue, forms of volatile memory store data temporarily so that the CPU can access it quickly. As processors improve, memory has continued to lag behind and cause bottlenecks to system performance. To alleviate this issue, clocking for the I/O data bus has increased with each generation of new DDR memories to obtain larger memory transfers in the same time frame. Though this proves effective for smaller burst lengths, larger burst lengths will be wasting overhead by transferring data to cache that will not be used. This likelihood increases as memory sizes continue to grow and multi-core systems become more prevalent. As a result, different computer system applications use different DRAM architectures. These designs specialize and vary in power consumption, throughput, and cost. The following sections explain the history of these designs and their weaknesses. Using this knowledge, later sections will look at solutions to the issues that currently plague memory designs.

History of DIMM Architectures

DRAM has been one of the fastest growing aspects of modern computing devices since its first forms in the 1960s. One of the first mainstream designs was the Fast Page Mode DRAM (FPM DRAM). This architecture involved keeping the row access strobe (RAS) active while continuously signaling the column access strobe (CAS) to read a series of data from the same row (shown in Figure 4). By keeping a row active in the MC, the latency needed to open that row

is induced once for every set of consecutive column addresses. Though simple in nature, this design provided significant speed increases with almost no added cost to the devices.

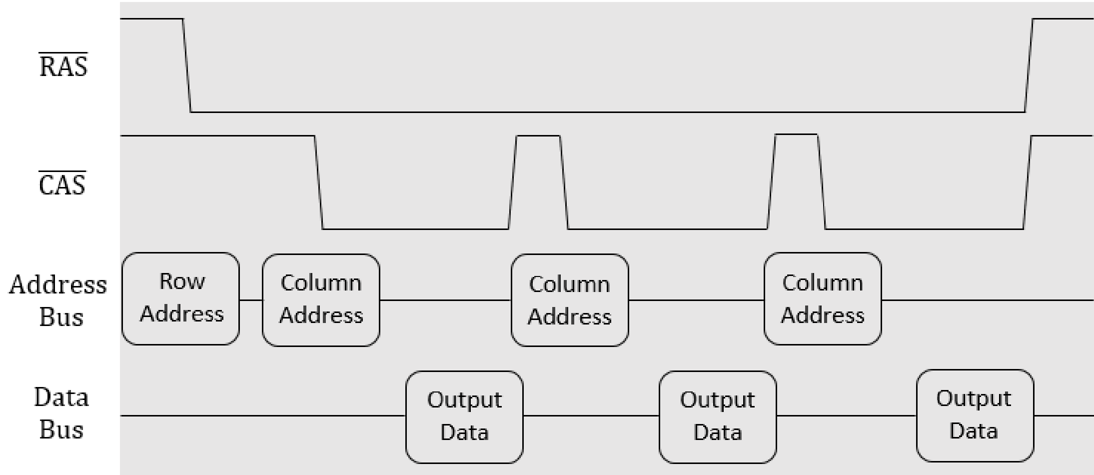

Figure 4: FPM DRAM read operation [1].

Following the era of FPM DRAM came Extended Data Out DRAM (EDO DRAM), which added a series of latches following each column multiplexer. This allowed the memory module to hold the data at the output pins while a concurrent column address travels across the address pins (shown in Figure 5). This slightly faster version started to take over FPM DRAM in 1995 and quickly led to the introduction of the Burst EDO DRAM (BEDO DRAM).

Similar to EDO DRAM, BEDO DRAM would burst four consecutive column addresses worth of data given the starting column (shown in Figure 6). This again increased the data transfer rate by incurring the CAS latency only once before data became available for each strobe of the CAS line. This method relies on the use of spatial locality to increase overall throughput of a particular process.

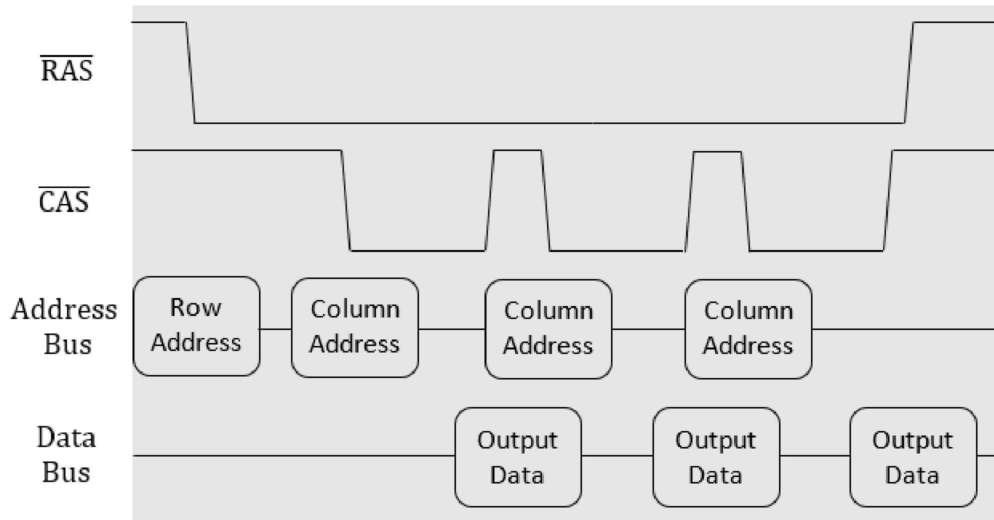

Figure 5: EDO DRAM read operation [1].

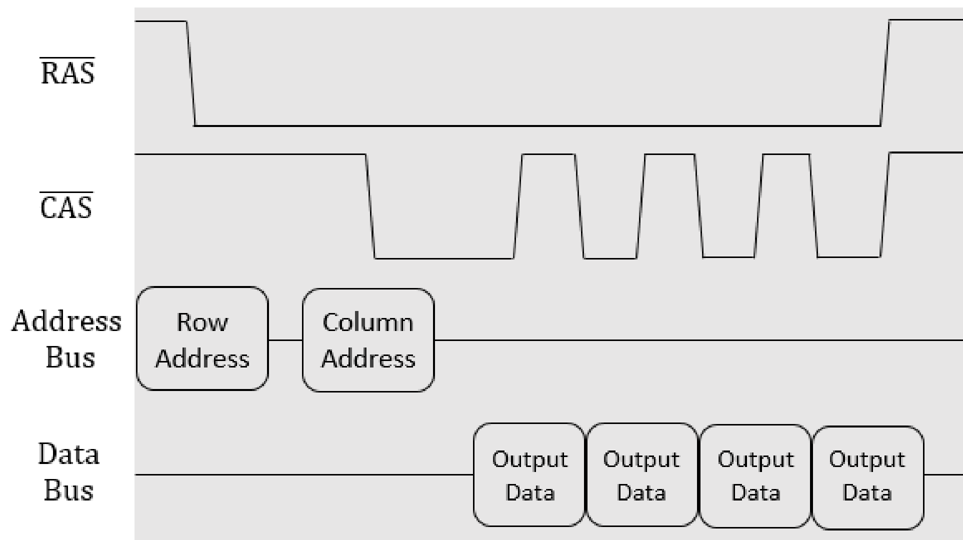

Figure 6: BEDO DRAM read operation [1].

As seen from the developments in DRAM that have been discussed so far, each new design is inexpensive to implement but yields tremendous performance advantages [1]. These low-cost changes have become increasingly rare since the introduction of Synchronous DRAM (SDRAM), which was in most modern computing systems by the turn of the 21st century. Unlike

its predecessors, SDRAM used a single-edged clock to synchronize the information that travels between the DIMM and the DMC (shown in Figure 7). Initial designs of SDRAM were slower than BEDO DRAM, but the introduction of a clock allowed reliable communication with the DMC. This led to the modern DDR DRAM architectures that currently dominate the market.

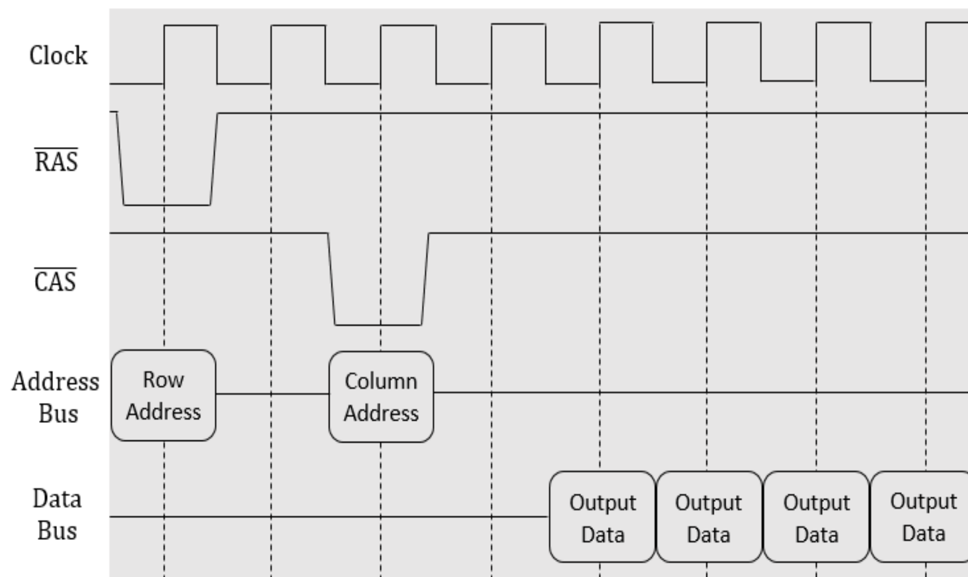

Figure 7: SDRAM read operation [1].

As the name suggests, DDR SDRAM differs from SDRAM by passing data on both the positive and negative edge of the clock cycle. This effectively doubled the rate of data transfer without having to double the clock speed. By using the positive and negative edge, DDR maintains low signal integrity requirements for the connection between the DMC and the DIMM. Since the introduction of DDR SDRAM in 2000, three more generations of DDR have each shown improvements in the data transfer rate. To achieve these improvements, engineers have created a standard architectural layout with specified timing parameters for communication. Using techniques like phased-locked loops (PLLs), engineers are able to ensure faster timing

with reliable accuracy. By clocking the I/O bus faster, and making the prefetch larger in each generation, engineers have been able to increase memory speeds despite traditionally slow MCs.

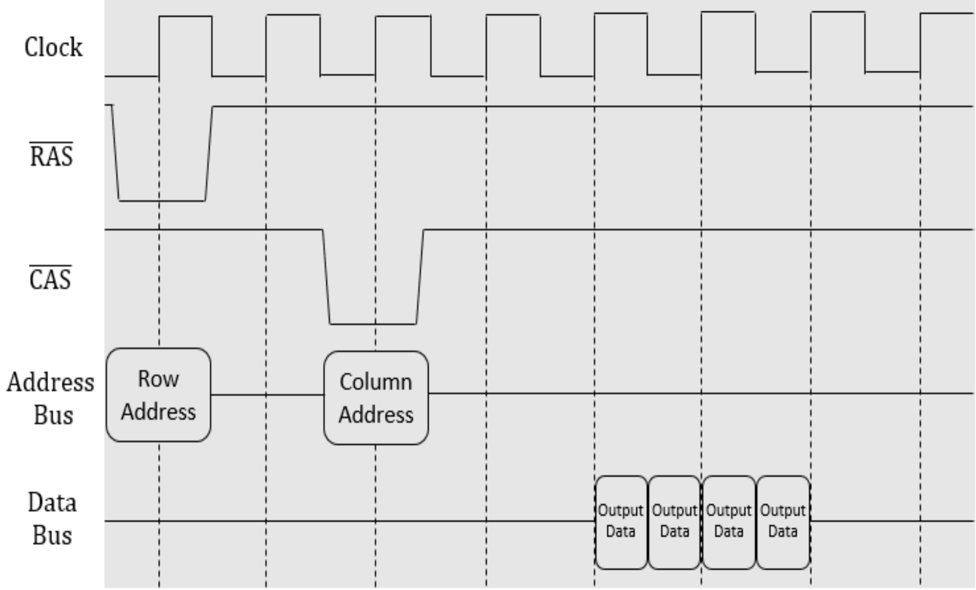

Figure 8: DDR SDRAM read operation [1].

Rambus DRAM

One of the main competitors to SDRAM in the 1990s was Rambus DRAM (RDRAM). Developed by Rambus Inc. for almost two decades, these memory modules had a unique narrow bus compared to the traditional large dedicated busses in SDRAM [2]. The simple RDRAM interconnect, between the DMC and the DIMM, communicated at significantly faster speeds. The smaller data bus, however, meant that address, data, and control information would all need to be share the same lines. To accomplish this, RDRAM uses a protocol similar to network packets, where DRAM chips are responsible for deciphering each packet. These complex DRAM chips not only increase cost but also create more complexities in the internal DIMM hardware.

Although certain system controllers, such as the Intel i850, used the Rambus architecture, it eventually faded out of production in favor of the lower cost DDR architecture [2, 3].

Graphics DRAM

Graphics DDR SDRAM (GDDR SDRAM) is a similar memory architecture to DDR, but specifically designed to communicate with the GPU. Instead of placing MCs on a DIMM, MCs are placed directly on the PCB surrounding a graphics processor. With a larger data bus than DDR interfaces, GDDR achieves higher throughput using a multi-channel architecture. Each modern GDDR5 DRAM MC provides up to 8Gbps [4]. By placing 12 MCs around the GPU, a traditional 6-channel GDDR5 configuration can achieve upwards of 84Gbps. Some graphics cards, such as the AMD Radeon R9 290 series, have up to 16 MCs placed around the GPU but only maintain about 80Gbps due to their slower 5Gbps MCs [5]. By using a wider bus, GDDR5 is able to alleviate the processing bottleneck needed for higher end graphics programs. Figure 9 shows a modern GDDR5 layout where each MC provides 32-bits to form 6 independent channels.

With the success of GDDR5 and the growing concerns over memory bottlenecks, JEDEC finalized a new standard called GDDR5X [6]. This memory standard doubled the prefetch size from 8n to 16n and effectively allowed 512-bit transfers per memory access. GDDR5X also contains MCs that have speeds up to 12Gbps [7]. Provided as a cheaper alternative to 3D stacked memories, GDDR5X can be found in graphics cards like the Nividia GeForce GTX 1080 Ti. To improve this technology, Samsung executives have proposed GDDR6, with up to 14Gbps per MC [8]. Though projected for a 2018 release date, new graphics cards often take longer than expected to reach the market.

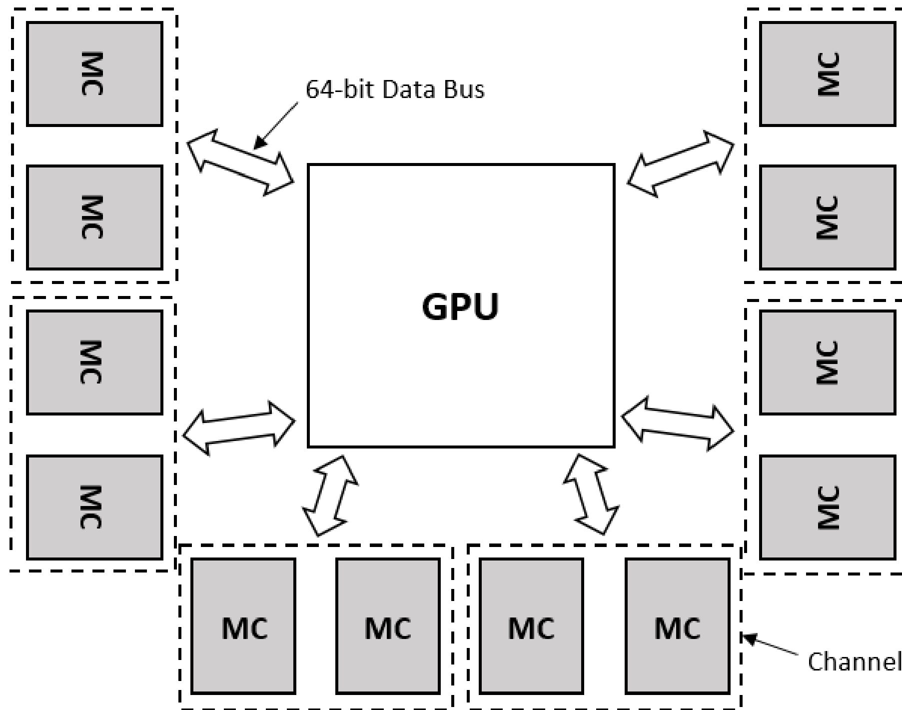

Figure 9: GDDR5 top view based on Nvidia's GeForce GTX Titan Black [9].

Current DIMM Issues

Scalability

Since the introduction of the first DDR memory module, engineers have increased transfer speeds by increasing the data rate for a DIMM's I/O bus. As a result, the internal DRAM hardware has seen few changes with new generations of DDR SDRAM (shown in Table 1). To hide the slow internal DRAM core, each generation of DDR has increased the prefetch length. This means more data returns for each column and row address that travels to the DIMM. For DDR4 modules, they use bank groups to divide the prefetch into two groups of eight. These bank groups act independently to achieve faster data rates than DDR3 while using the same slow DRAM cores [10, 11]. Through this technique has been successful in the past, larger prefetch architectures will not only be difficult to achieve, but will likely transfer data that goes unused.

Table 1: DDR MC clock versus I/O data bus clock [11, 12, 13, 14].

DDR Generation	Internal Memory Chip Clock (MHz)	I/O Data Bus Clock (MHz)
DDR	100 – 200	100 – 200
DDR2	100 – 266	200 – 533
DDR3	100 – 266	400 – 1066
DDR4	133 – 266	1066 – 2133

Pin Count

DRAM cells have decreased in price since the introduction of DDR SDRAM. This allows engineers to pack more memory cells into DIMMs without creating large price tags. Packaging costs on the other hand, are not decreasing at similar rates. This means that pin count has become a critical contributor to the manufacturing costs of DIMMs [1]. In current architectures, reducing pin count to reduce costs would decrease memory bandwidth. However, pin count also plays a large role in power consumption and heat dissipation.

Heat and Power Consumption

DIMMs are not designed to consume significant amounts of power or dissipate a lot of heat. In fact, they are considered more simplistic hardware devices because they rely heavily on the DMC to tell them what actions to perform. DIMMs will consume more power and dissipate more heat as they become larger and their hardware becomes more complex. Additionally, Fully Buffered DIMMs (FB DIMMs) also increase heat dissipation due to the extra registers required to buffer incoming data [1]. By increasing DIMM sizes, the number of FB DIMMs, and faster I/O clocking, DRAM systems become large contributors to power consumption and heat dissipation. Growing power and heat issues threaten system designs like blade servers, which rely on shared cooling and power devices within an enclosure.

Refresh Rate

As DIMMs continue to grow in size, they are capable of storing significant amounts of memory. According to JEDEC standards, rows containing memory need to be refreshed every 64ms to prevent data loss (32ms for temperatures over $85^{o}C$) [15, 16]. That means that larger memories will need to spend more time refreshing rows in the same time frame. With increasing chip density and more rows in a refresh bundle, the time spent refreshing cells becomes a significant problem to memory performance. This will be counterproductive to the advancements made in data transfer rates with future DDR generations. Figure 10 shows the proportional relationship between the capacity of the memory and the time spent refreshing cells. Note that values are doubled when the temperature is above $85^{o}C$.

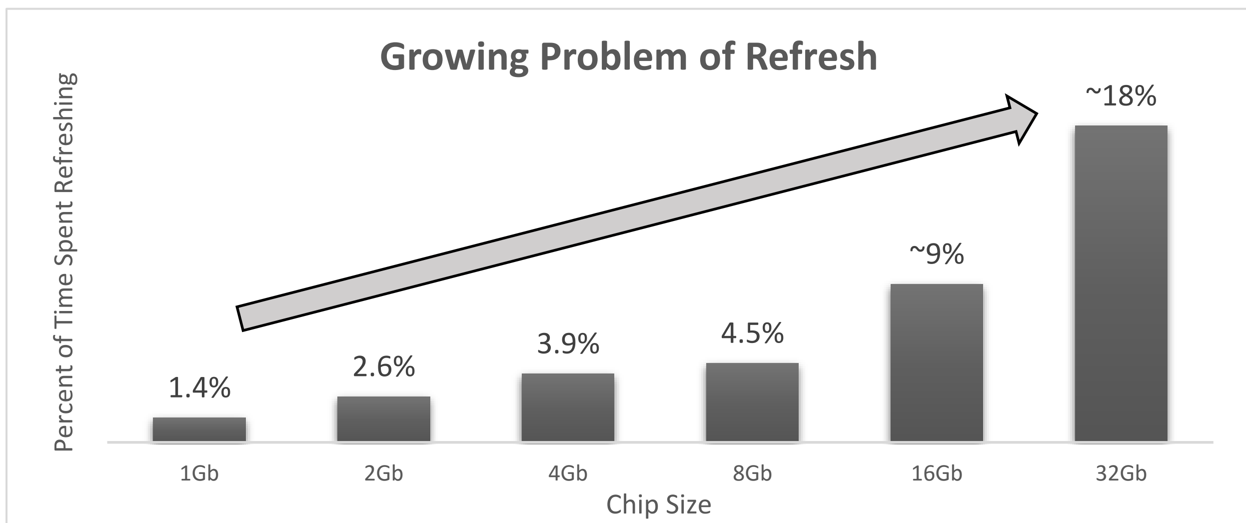

Figure 10: Time spent refreshing memory for temperatures below $85^{o}C$ [16].

Commodity Status

DIMMs have been marked as a commodity compared to other components within a computer system. As a commodity item, cost plays a major factor when considering different DRAM architectural decisions. This can affect memory speeds as engineers trade off the complexity, power consumption, and the effected cost. In order to make a new architectural design competitive in today's market, the designer needs to prove that the faster speeds are worth the higher price tag.

3D Stacked Memory

To alleviate many of the issues discussed above, industry leaders have developed new DRAM architectures involving stacked MCs. Unlike traditional DDR memories, where each MC connects directly to the PCB, 3D memories involve placing MCs on top of each other to create a "stack". This allows similar memory sizes to take up significantly less form factor and communicate with each other using through-silicon vias (TSVs). By using TSVs instead of the traditional pin connections, 3D stacked memories have larger bus widths, and higher channel counts, to achieve higher throughput.

High Bandwidth Memory

High Bandwidth Memory (HBM) is a version of 3D stacked memory developed by AMD and Hynix. Designed to communicate with GPUs, it targets high throughput by using a 1024-bit wide data bus spread across 8 channels (128 bits per channel). By stacking the chips, 1GB of HBM takes approximately 35mm^2 of space compared to 672mm^2 in a traditional GDDR5 architecture [17]. With a smaller footprint, these 3D stacks are placed on an interposer with the processor (shown in Figure 11). The close proximity, higher chip count, and wide data bus allow HBM to achieve over 100GB/s per stack [17, 18, 19].

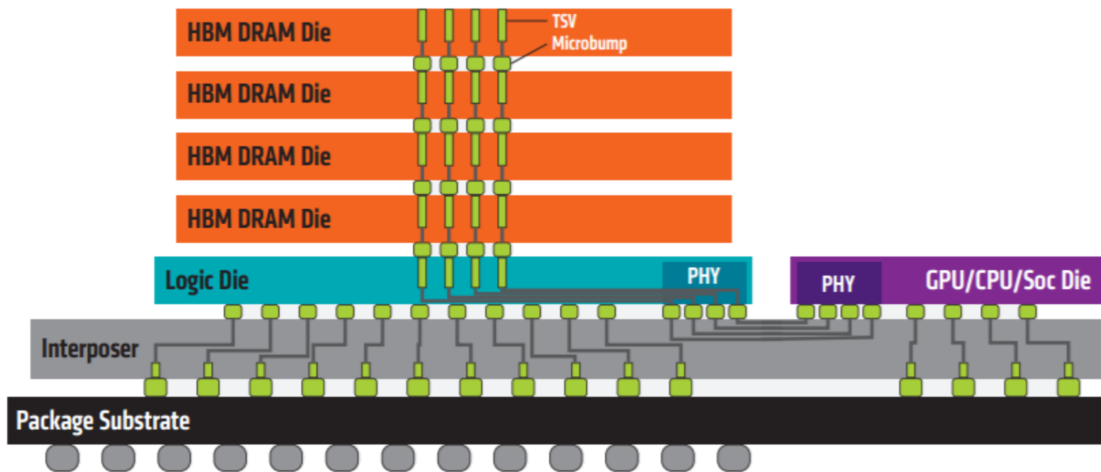

Figure 11: HBM cross sectional view [17].

HBM had limited shipping with AMD's Radeon R9 Fury, Nano, and Fury X graphics cards before developments quickly began on its replacement HBM2. Unlike its predecessor, HBM2 can stack up to 8 dies as opposed to HBM's 4. HBM2 also introduced pseudo channels to break individual channels into two sets of sub-channels (shown in Figure 12). In what is called legacy mode (mode used in HBM), each read/write command will transfer 256 bits during 2 cycles. Using pseudo channel mode (mode used in HBM2), the "128-bit bus is split into 2 individual 64-bit segments" [20] that still transfers 256 bits but reduces the four-bank activation window time [1, 20]. In summary, HBM2 provides up to 256GB/s per stack compared to HBMs 128GB/s and can offer up to 8GB per stack compared to HBMs 4GB [21, 22]. The recent Nvidia Quadro GP100 graphics card incorporated 4 stacks of HBM2 with a memory sizes of 16GB. AMD's Vega graphics cards plan to hold HBM2 as well, when they are released in the coming months of 2017.

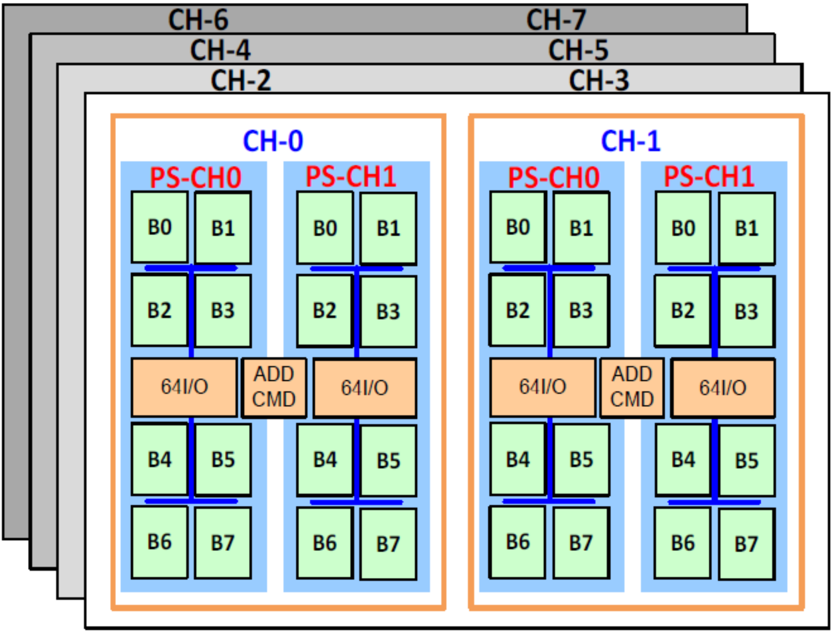

Figure 12: HBM2 pseudo channels overview [21, 22].

Continuing their push forward, industry leaders plan to release HBM3 between 2019 and 2020. With up to 16 stacked dies and 16GB per stack, HBM3 promises to double the data transfer rate to 512GB/s per stack [23]. These advancements in 3D stacked memory pose future power issues as stacks grow in size and more TSVs are used to increase bandwidth. To avoid the potential power and bandwidth walls, this paper proposes the use of 3D ring based SWOs to achieve an efficient communication scheme. The results show that using this scheme can reduce TSVs and power consumption while maintaining similar speeds to current HBM standards.

Wide I/O Memory

Wide I/O Memory is a 3D stacked memory backed by Samsung that specializes in low power applications. Designed with the mobile market in mind, Wide I/O originally promised stacks of DRAM to be place directly on top of the processor. This engenders significant thermal and power issues to ensure that the processor heat does not cause memory failure. Instead, most

architectural layouts still use an interposer like HBM. Wide I/O promised up to 17GB/s, but failed to make it into production [24].

Wide I/O 2 Memory builds on its predecessor by improving bandwidth to 68GB/s and minimizing power consumption. Unlike the 512-bit data bus used by Wide I/O, Wide I/O 2 increases the bus width to match HBM at 1024 bits [25]. Targeting high-end low power applications, Wide I/O 2 became a JEDEC standard in 2014. To date, this 3D stacked memory has yet to make it to production, partly due to its high price tag for a mobile market.

CHAPTER II

RING BASED MEMORY

Over the past two decades, researchers have proposed many solutions to solve the current memory bottlenecks. The developments of HBM and WideIO promise significant upgrades from the current DIMM speeds but engender new issues. One issue is distributing a low skew clock, to all DRAM dies, to increase speeds with minimal power consumption. Another problem is that refreshing memory continues to increase latencies and causes longer queue lengths in the DMC. Finally, TSVs have limited clocking speeds and 3D memories must rely on higher channel counts to achieve higher throughput. In order to increase speeds in future 3D memory generations, research papers have explored solutions to solve these issues.

Skew Cancellation for HBM

A paper released last year by Ahn and Yoo, from Hanyang University, explored a solution to reduce skew between 32 DQ lines used for the data bus [26]. Increases in skew will reduce the window for valid data and result in decreased bandwidth. To achieve this lower read skew rate, they use buffers on the logic die that are adjusted by successive approximation register logic. For write skew, they form loopback paths on the logic die by controlling the buffer delays. After cancellation, they are able to achieve 15 times less skew with a difference of 18ps still left between the DQ lines [26]. This is an important aspect of HBM as less skew will result in faster clocking and therefore faster memory speeds.

Massed Refresh

Another paper published by Thakkar and Pasricha from Colorado State University proposes a technique to reduce refresh time for Hybrid Memory Cubes. This is another form of 3D stacked memory that is developed by Intel and other memory based companies. The paper produced from Colorado State suggests using subarray-level and bank-level concurrency to reduce the time spent refreshing cells [27]. By mapping multiple DRAM rows to subgroups, which are refreshed concurrently, significant time is saved refreshing the memory. This reduces the power consumption by 5.8% and improves throughput by 6.3% [27].

Simultaneous Multi Layer Access

A technical report published by professors at Carnegie Mellon University proposed an IO organization to increase memory bandwidth [28]. They used the internal bandwidth of multiple layers to increase IO speeds. By time-multiplexing across DRAM dies, the professors claim up to 55% improvements in performance for multi-programmed workloads. The most important factor, however, is that it maintains low costs by leveraging internal global bit-lines instead of adding external global bit-lines [28]. The report was published in 2015 and performance measurements are based on HBM standards instead of the higher bandwidth HBM2.

Ring-Based Interconnect

As mentioned in the previous summaries on 3D stacked memory research, there are a number of proposed techniques to improve the communication between DRAM chips. In this paper, the focus is on ring-based SWOs to provide a low skew, fast clock to all memory layers. The SWO uses two clock wires connected on one side by a Mobius Crossing and an inverter pair (shown in Figure 13). Extraction points along the ring contain differential amplifiers (clock

recovery circuits) to obtain a clock from the standing wave. Because of the Mobius Crossing, a virtual crossing exists at the midpoint of the ring opposite the inverter pair. The differential signal seen near this virtual crossing is not large enough to extract a fully differential clock. This creates a "dead-zone" where clock recovery circuits cannot exist. As the resonant ring gets faster, the size of the "dead-zone" increases. The virtual crossing also creates a phase change, where the positive and negative inputs to the differential amplifier flip to ensure all DRAM dies see the same rising edge.

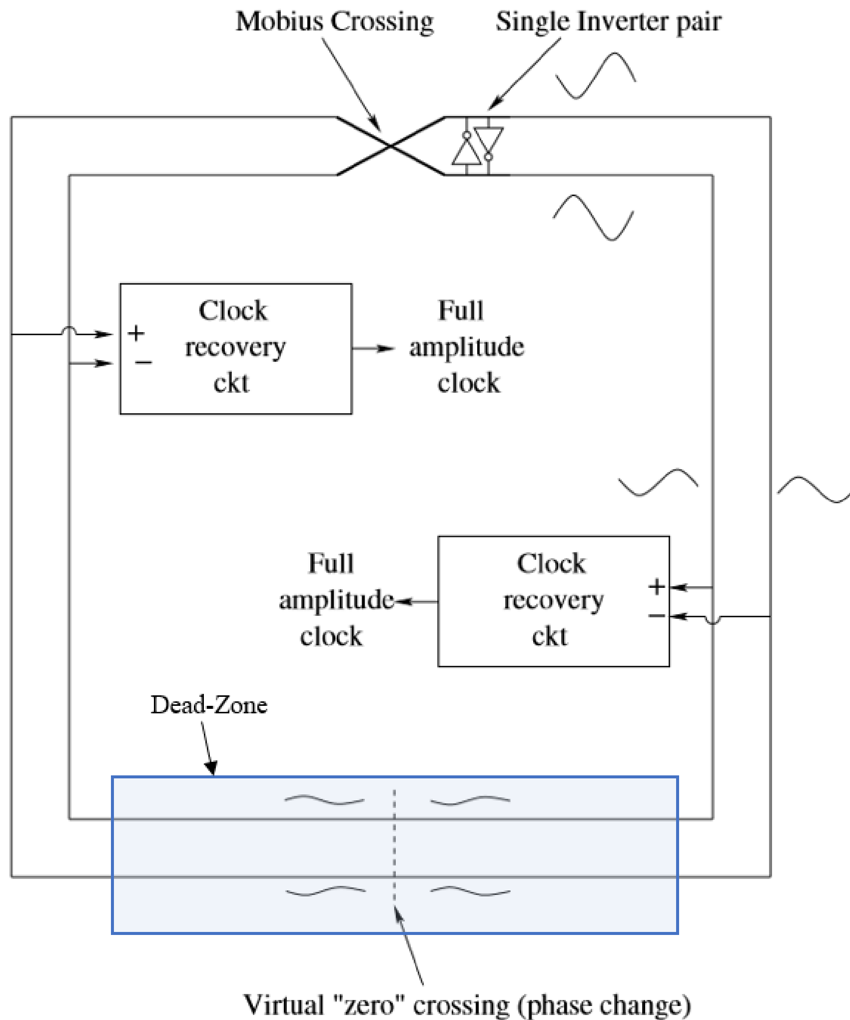

Figure 13: Standing wave ring clock [29, 30].

The memory architecture using a ring-based topology (MART) utilizes both a 3D resonant SWO and a series of data lines. These wires intersect each channel to form insertion-extraction stations (IE stations), where data is injected or removed from the rings. To ensure that all IE stations receive a full amplitude clock, and no station falls in the "dead-zone", longer traces are placed at the top of the stack (shown in Figure 14 right). This increases the total length of the SWO, but maintains minimal skew compared to traditional clocking schemes. Because of the increased SWO length, the TSVs used for the ring need to be wider to reduce resistance. This allows the ring to oscillate at faster speeds despite the increased length. Figure 14 shows a traditional HBM stack (left) compared to a MART stack (right) where the dotted line shows the division between channels on the same DRAM die. The 212-bit bus for the traditional HBM stack represents all lines for a single channel [20].

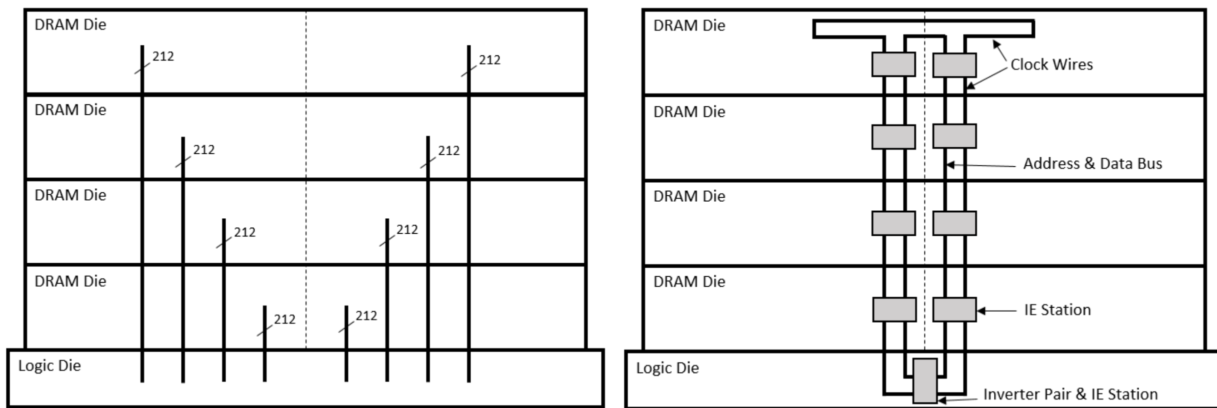

Figure 14: Cross sectional view of HBM (left) and MART (right) for a 4GB stack.

With each rotation of the SWO, addresses and data travel between IE stations. Because there are 9 total IE stations, the clock needs to oscillate at 18GHz to provide a round trip time similar to traditional HBM TSVs (0.5 nanoseconds). Each IE station is equidistant apart to

ensure that information arrives within the desired period. The stations contain a series of flip-flops to buffer the data and address lines between DRAM chips. The differential amplifiers, that extract the full amplitude clock from the ring, will drive the clock input to these flip-flops. A series of multiplexers and asynchronous FIFOs are used in each IE station to communicate between the channels and the 3D data rings. This is an area of ongoing research and this paper assumes that the IE stations operate at the desired ring frequency.

Control lines and the address bus also utilize the ring and the IE stations. This allows the address and die selection to move consistently with the data during write commands. The data is then stored temporarily in the IE station while the bit-lines and corresponding row activate to write the data. Three control lines are added to select which of the 8 channels the address/data corresponds to. By sharing the address and control lines, the number of TSVs decreases to reduce power consumption and manufacturing costs.

By using the energy recycling capabilities of the ring, this high frequency clock will consume less power while maintaining high transfer rates. The ring relies on parasitic capacitance and inductance to operate. To create a predictable environment for the clock, shield planes are used to provide the dominate ground capacitances in relation to the clock wires. It is important to note that the 3D ring is still a TSV, where the cylindrical clock wires will travel through the silicon of each die to connect each section together. The Mobius Crossing and inverter pair are located on the logic die to allow the longer traces at the top of the stack to accommodate the "dead-zone". Figure 15 shows the dimensions of the ground plane and clock TSVs to achieve speeds of 18GHz.

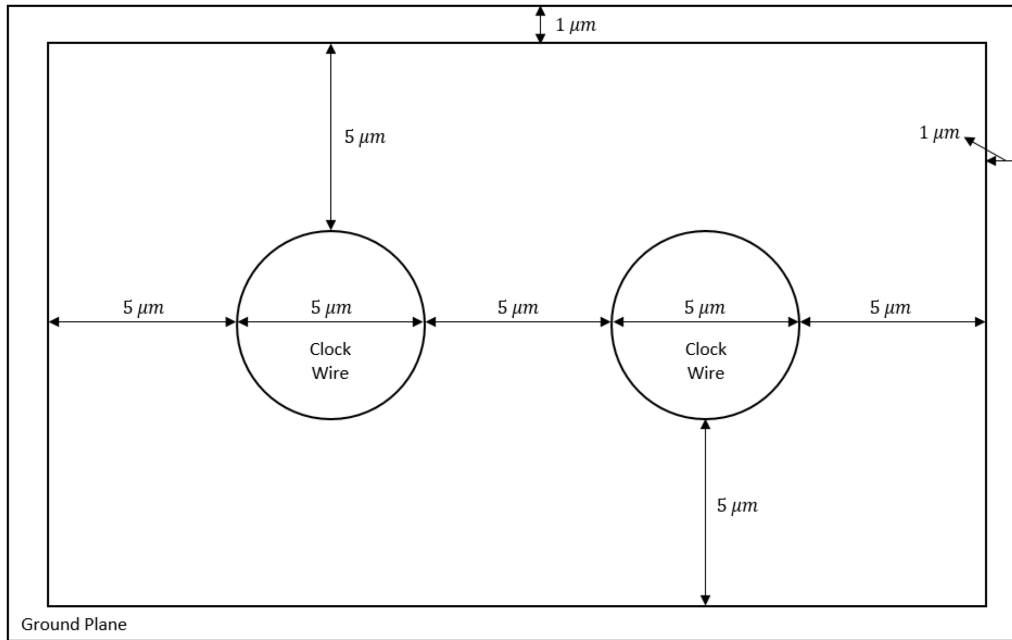

Figure 15: Cross-sectional view of 3D clock wires.

For 8GB HBM stacks, manufactures grind the substrate down to maintain similar heights to 4GB HBM stacks. Each DRAM die in an 8GB stack represents an independent channel to create an 8-channel architecture similar to the 4GB stacks. The channels, however, are larger and consist of 1GB each compared to 512MB. These larger stacks typically only exist in servers and high performance computers. Table 2 shows the heights of for all HBM stacks according to the JEDEC standard while Figure 16 shows the traditional HBM configuration and the MART configuration for 8GB stacks.

Table 2: HBM stack heights [19].

Stack Size	Minimum (μm)	Typical (μm)	Maximum (μm)
2GB / 4GB / 8GB	695	720	745

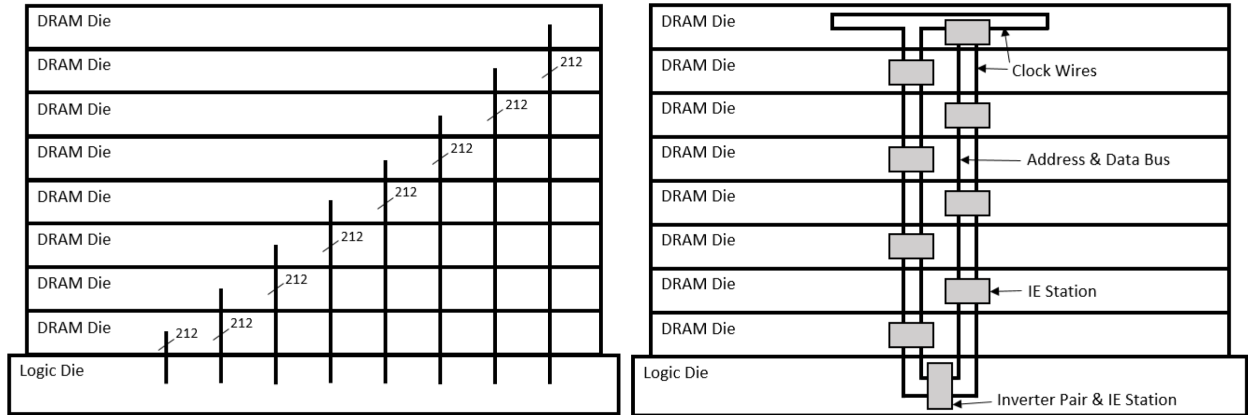

Figure 16: Cross sectional view of HBM (left) and MART (right) for an 8GB stack.

CHAPTER III

SIMULATION RESULTS

The (MART) was tested using three main simulation tools. HSPICE provided electronic simulations to ensure the clock operated at the desired frequency. Raphael produced the capacitance, inductance, and resistance extraction for the clock wires to operate. Finally, using the timing parameters obtained from the previous two simulation tools, Ramulator provided DRAM simulations using CPU traces [31]. Ramulator is a C/C++ open source program provided by the SAFARI group at Carnegie Mellon University. It has pre-built DRAM models for DDR3, LPDDR3, DDR4, LPDDR4, GDDR5, WideIO, WideIO2, and HBM. Though this program is only an estimate of memory performance, it presents useful insight into the operation and utility of different memory architectures.

16nm Process

The simulations performed in this paper used the 16nm PTM fabrication process provided by Arizona State University [32]. Before simulating the ring, the high-to-low and low-to-high propagation delays were measured for a minimum size inverter in an inverter chain. By varying the width of the PMOS in the inverter, a width of 44nm was chosen to equalize these delays. This was critical to ensure that the inverter pair in the ring switches at the same rate for each clock rotation. Figure 17 shows the HSPICE results with the NMOS length at 16nm, the NMOS width at 32nm, the PMOS length at 16nm, and the PMOS width varied from 20nm to 80nm.

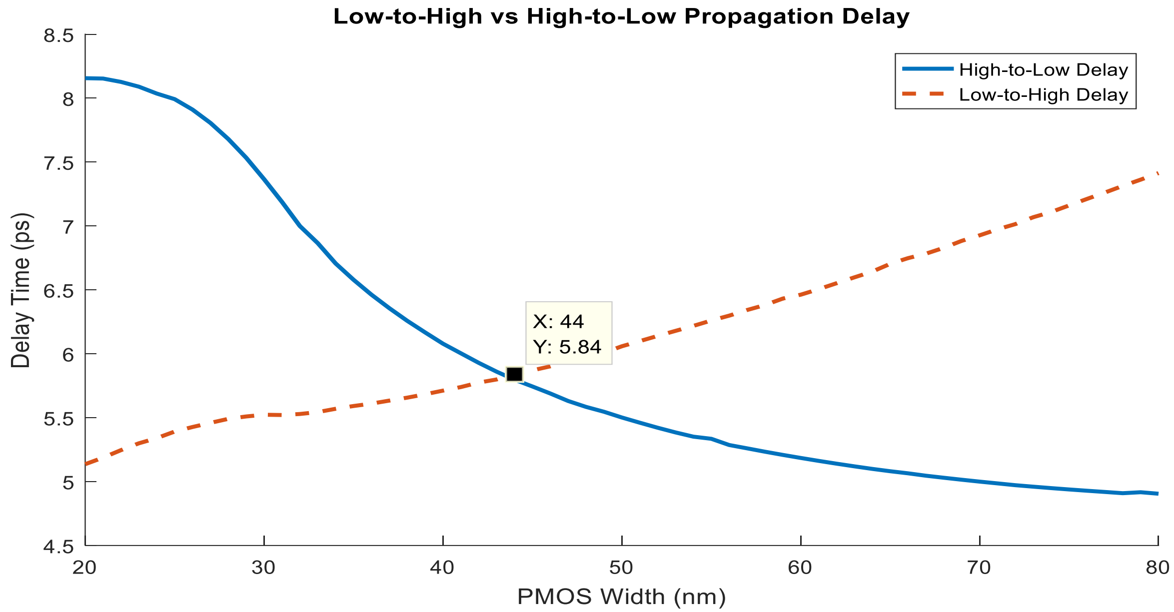

Figure 17: 16nm process delay equalization.

3D Ring Clock

In order for the ring to operate, the capacitance, resistance, and inductance of the clock wires were extracted using Raphael. After obtaining the desired values, a distributed model of the clock wires was constructed in HSPICE. This distributed model utilized 24 individual T-sections that evenly distributed the resistance, capacitance, and inductance across the length of the ring. As mentioned in Table 2, the typical height of a 2GB, 4GB, and 8GB HBM stack is 720µm [19]. Thus, the length of the ring must be at least 2200µm for two reasons. First, the ring is double the height of the stack because it must connect to all 8 channels. Second, the extra trace at the top of the stack must be at least 33% of the total ring length to accommodate the "dead-zone". This left 1474µm of the ring capable of extracting a full differential clock, which meets the requirement for the stack height.

Because 9 IE stations exist, the clock simulated in HSPICE was adjusted to run at 18GHz. Clock extraction points were placed at the 16 T-sections closest to the inverter pair. These 16 T-sections also included differential amplifiers to extract a full differential clock signal with minimal skew. These differential amplifiers use a DC bias transistor and a current mirror active load to provide a single-ended output driven by two inverters (shown in Figure 18). The size and number of these inverters can vary based on the number of flip-flops they drive in each IE station.

Figure 19 shows the 16 extracted points along the SWO that receive a full differential clock. The period of the extracted clock is 55.79ps, which equates to 17.92GHz. The rising skew between the 16 points is 2.54ps, while the falling skew is 3.55ps. This is 5 times less than the skew minimization technique used by Ahn and Yoo [27]. Based on these results, the SWO can deliver a clock to all IE stations and maintain a round trip time of 0.5 nanoseconds. This allows comparable speeds to current HBM standards but utilizes fewer TSVs.

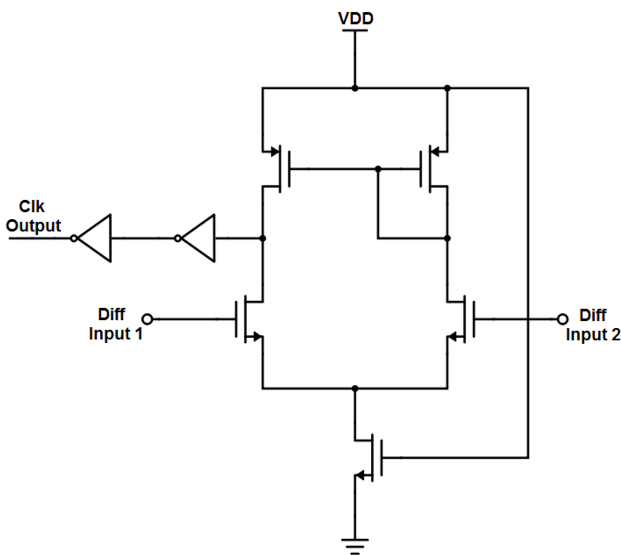

Figure 18: Differential amplifier circuit to extract clock signal [29, 30].

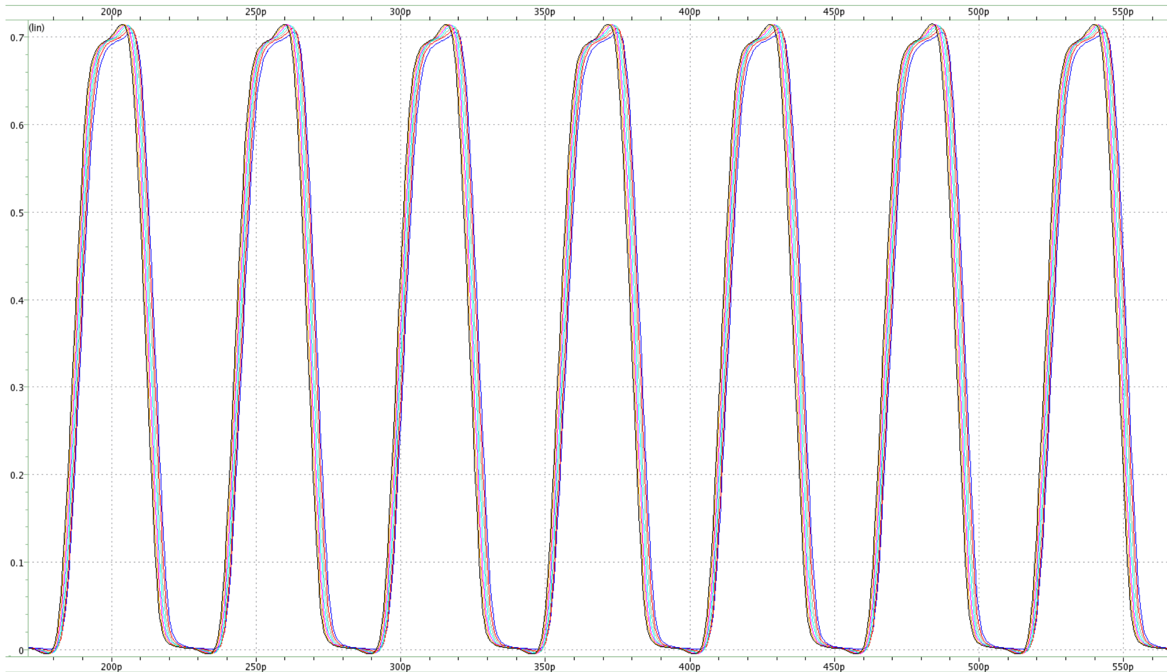

Figure 19: 18GHz extracted ring clock signal at 16 different locations.

Current DRAM Speeds

Using the built-in models from Ramulator [31], modern DRAM architectures were simulated with 9 different CPU traces. Each DRAM architecture was run using multiple channels to accurately represent the memory in its fastest configuration. DDR memories had a maximum of 8 ranks per channel, since capacitance loading on more ranks is not feasible [33]. The 3D stacked memories, HBM and WideIO2, had 8 channels based on the JEDEC standard [28, 29]. Every DRAM chip was run with the maximum clock speed available to show the fastest speeds the memory configuration was capable of. HBM was run with only 4Gb since 8Gb was not available in the Ramulator package. Table 3 shows an overview of the configuration settings for each DRAM architecture.

Every DRAM configuration was run twice, once with the cache enabled (to simulate an accurate computer system) and once with the cache disable (to exercise the DRAM architecture).

The CPU traces used came from the SPEC CPU 2006 benchmarks and contained a variety of trace sizes and memory requests. Table 4 shows the total number of memory requests as well as the percentage of those requests that were write commands. The final column marks the percentage of the read and write requests that were caught by the cache.

Table 3: Ramulator DRAM simulation configurations.

DRAM Architecture	# of Channels	Ranks per Channel	I/O Bus Speed (MHz)	Memory Size
DDR3	2	8	2133	8Gb
LPDDR3	2	2	2133	8Gb
DDR4	4	8	2400	8Gb
LPDDR4	2	2	3200	8Gb
GDDR5	6	1	7000	8Gb
WideIO2	8	2	1066	8Gb
HBM	8	2	1000	4Gb

Table 4: SPEC CPU 2006 trace statistics.

CPU Trace	Total Number of Requests	% of Requests that are Write Commands	% Requests that are Cache Hits
429.mcf	16965722	9.93	32.8
470.lbm	9988010	42.8	41.9
450.soplex	8748061	35.1	73.7
459.gemsfdtd	6918110	26.5	18.4
462.libquantum	6447884	16.3	3.1
437.leslie3d	4563215	25.2	11.2
433.milc	4333369	26.1	3.06
471.omnetpp	3984337	2.89	97.9
483.xalancbmk	3772390	2.55	95.8

Note: Cache size was 1MB, with a block size of 64 and an associativity of 8.

Based on the configurations shown in Table 3, Ramulator provides the theoretical maximum bandwidth that is achievable. This parameter is typically marketed by memory manufactures to show the potential speeds of different DRAM architectures. Though a configuration may have higher theoretical bandwidth, it does not guarantee better performance for all applications. Table 5 shows the theoretical maximum bandwidth for each configuration that was run with Ramulator. Though GDDR5 has the highest theoretical bandwidth, the results show that it is not necessarily the fastest architecture when running different CPU traces.

Table 5: Ramulator theoretical maximum bandwidth.

DRAM Architecture	Maximum Bandwidth (GBps)
GDDR5	336
WideIO2	136
HBM	128
DDR4	76.8
DDR3	34.1
LPDDR3	34.1
LPDDR4	25.6

The different DRAM architectures were tested against the 9 different CPU traces using the configurations in Table 3. The first important statistic examined from the results is the latency seen by each command that travels from the DMC to the DRAM. Though this latency can measure the average time a single command takes to execute in memory, it does not accurately represent the parallelization of multi-channel architectures. Instead, a clear line separates the low power memory options (LPDDR3, LPDDR4, and WideIO2) from the

traditional DRAM architectures (DDR3, DDR4, HBM, GDDR5). This shows that the low power memory options use slower DRAM cores to reduce power consumption. Despite the separation between lower power and traditional memories, there is no clear distinction between the memory architectures for different traces. Figures 19 and 20 show the average latency per command without and with a cache.

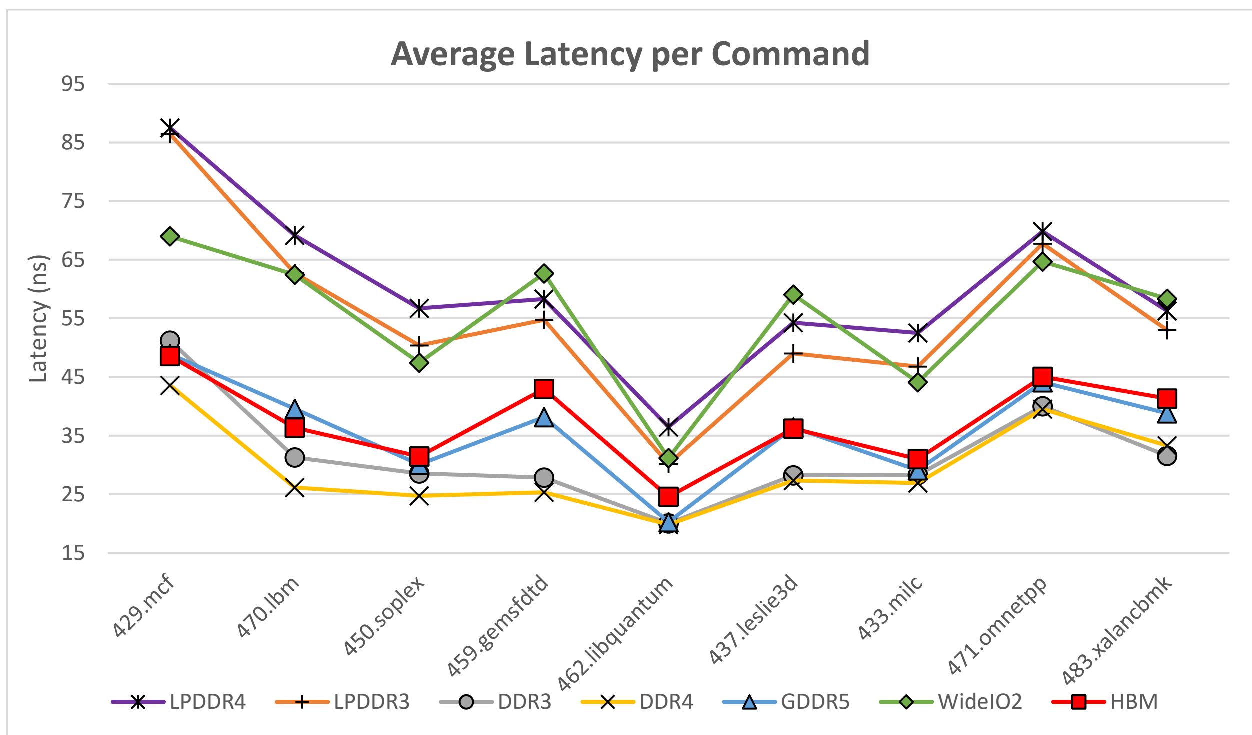

Figure 20: Average latency per command without cache (Appendix, Table 9).

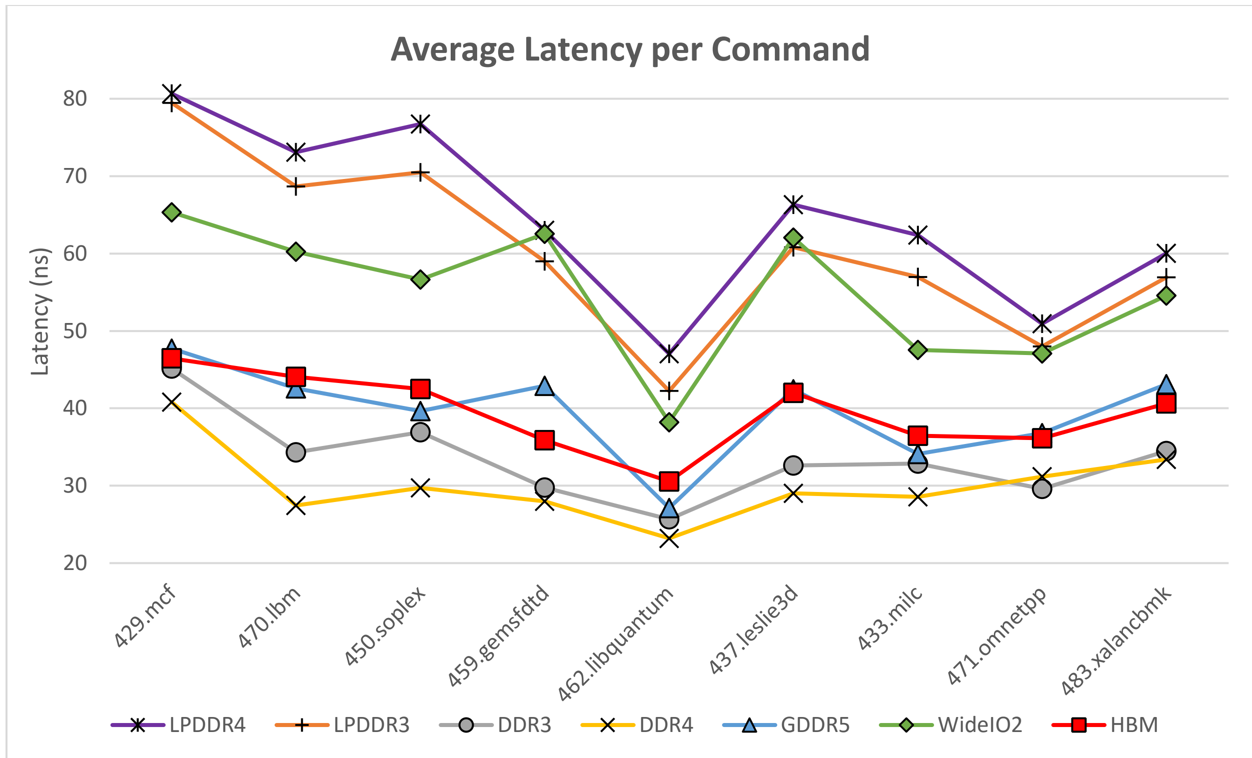

Figure 21: Average latency per command with cache (Appendix, Table 10).

Though the average latency per command is a useful statistic to measure the speeds of the DRAM core, it does not accurately represent the parallelization utilized by multiple channels. Measuring the total latency of all read commands over a CPU trace shows the differences between the different memory configurations. The configurations that utilize more channels and ranks resulted in significant speed increases over the lifetime of the program. Those that had at least 4 channels were approximately 50% faster than those with 2 channels. Traces 471.omnetpp and 483.xalancbmk had very small latencies because of their high cache hit rate. The results in Figures 22 and 23 show the total latency in milliseconds. It is important to notice that HBM performed better than all memory architectures for every CPU trace.

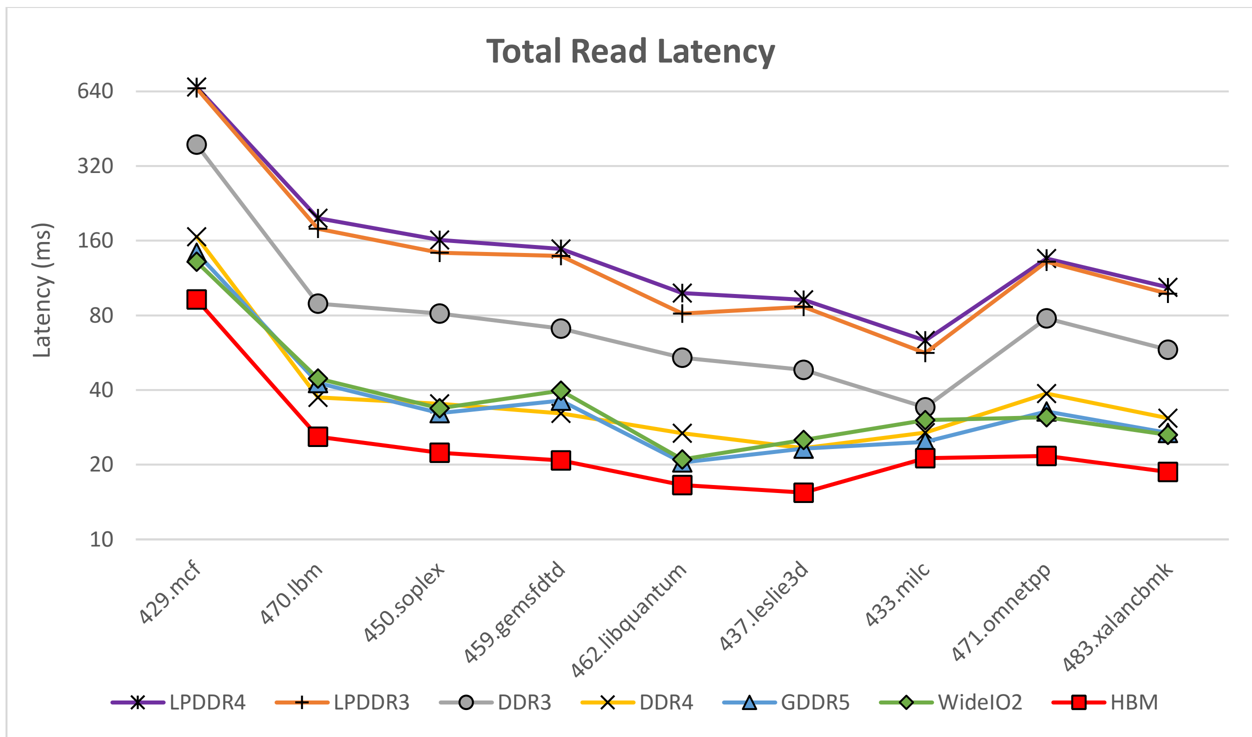

Figure 22: Total read latency without cache (Appendix, Table 11).

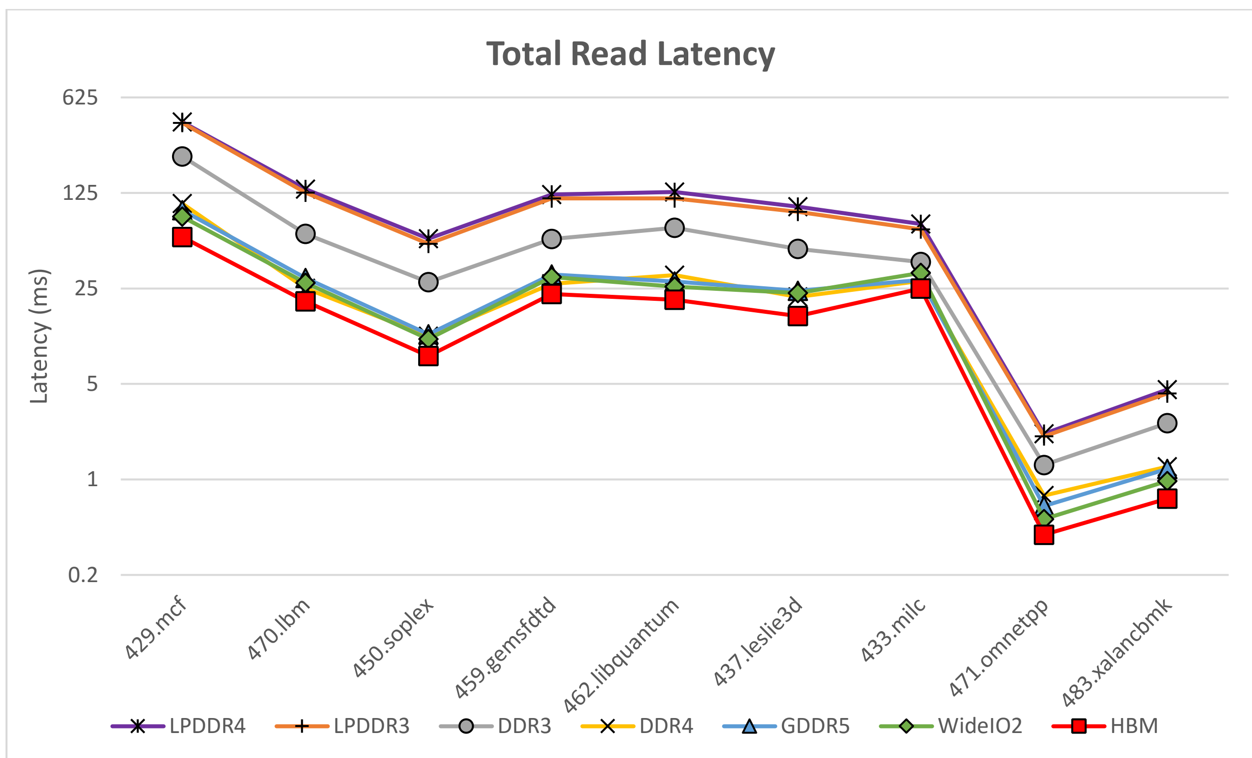

Figure 23: Total read latency with cache (Appendix, Table 12).

An accurate representation of the parallelization achieved by using multiple channels is the average queue length in the DMC. This statistic shows that the channel count is proportional to the average length of the queue, since commands are divided amongst different channels when possible. The longer the average queue length, the longer the latency for the command to execute in memory. Figures 24 and 25 show the average number of commands waiting in the queue per channel. The number of commands waiting idle grows as more commands are issued to the DRAM. This explains the higher average queue length for traces that have more memory commands compared to those with a higher cache hit rate (471.omnetpp and 483.xalancbmk).

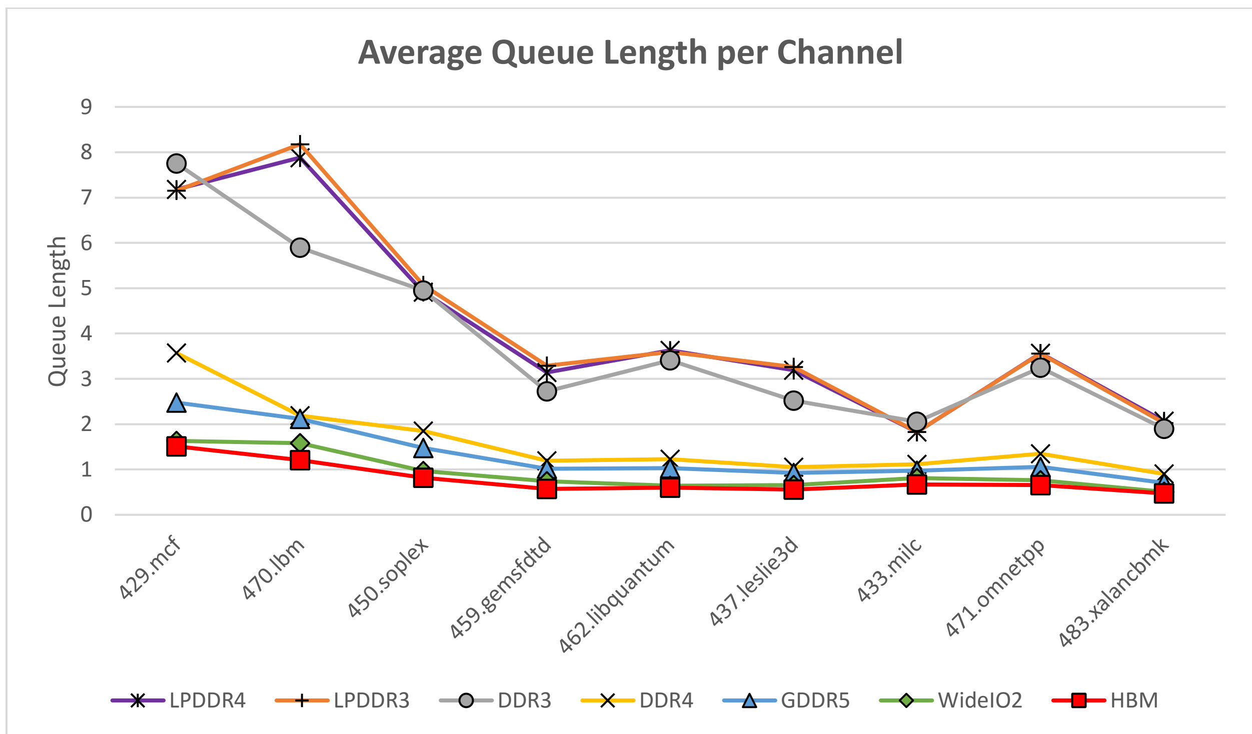

Figure 24: Average queue length without cache (Appendix, Table 13).

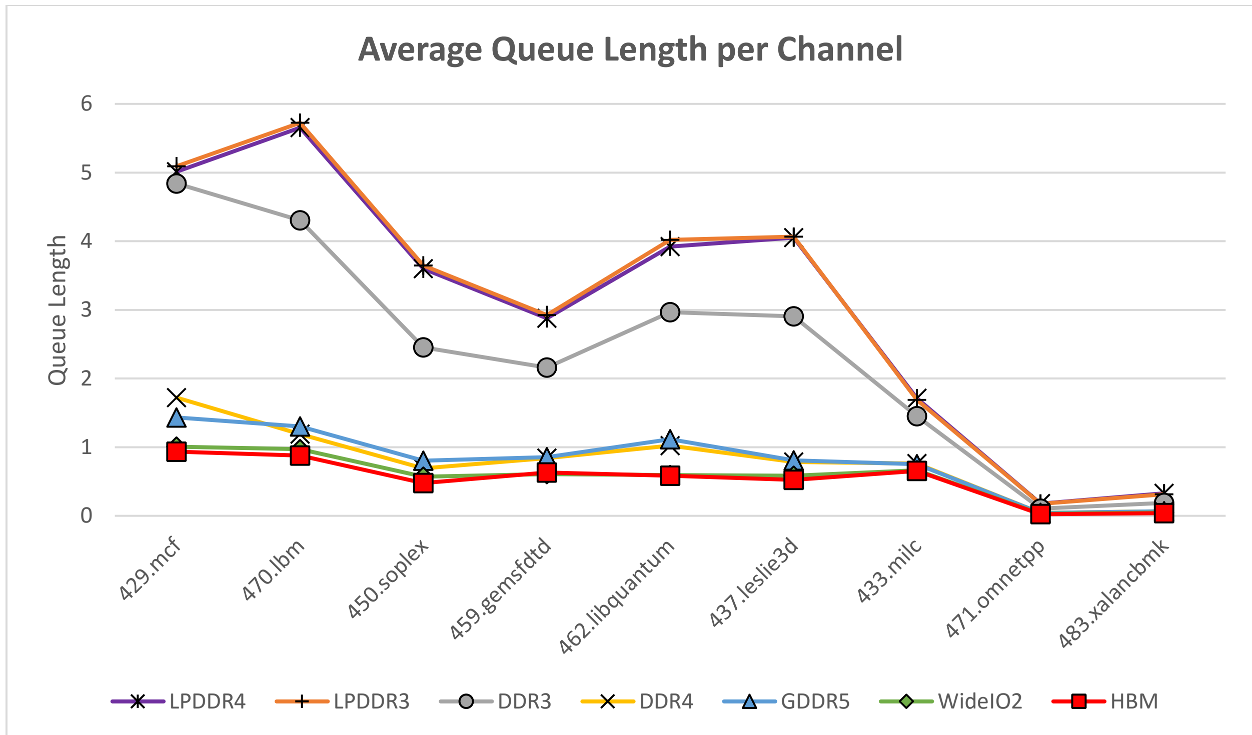

Figure 25: Average queue length with cache (Appendix, Table 14).

Consistently, in the results presented, HBM was faster than other DRAM architectures for all CPU traces. The increases in speed were more for larger programs, such as 429.mcf, compared to smaller programs, like 471.omnetpp. The results prove that the higher channel counts of 3D stacked memories like HBM and WideIO provide larger throughput compared to traditional DDR architectures. Because HBM proved faster than any other memory configurations, it is the basis for comparison with the MART.

Low Power Ring Memory

Using the HBM model as a template, the MART model was adjusted to incorporate the shared data bus. In the low power MART (LPMART) scheme, all 8 channels share the same 64-bit data bus and 14-bit address bus (this is based on a WideIO type interface). This makes a total of 156 TSVs for the address/data bus in Figures 14 and 16 (78 on each side of the dotted line).

Adding the clock wires and the control lines, the total number of TSVs needed for the LPMART is 296. Table 6 shows the TSV comparison between a traditional HBM and the LPMART scheme.

Table 6: Total number of TSVs for HBM and LPMART (assuming 8 channels) [20].

TSV Function	HBM	LPMART
Data	1024	128
Column Address	64	16
Row Address	48	12
Clock	16	4
Control Lines	544	136
Total	1696	296

HBM TSVs have an average height that goes through 2.5 DRAM/Logic dies for each clock cycle (for a 4GB stack). The LPMART only goes through 1 die per clock cycle because there are registers in every IE station. This gives an approximation for the capacitance each TSV experiences per clock cycle that can be applied to the following equation.

$$P_{dynamic} = C\, V_{DD}^2\, f_{sw}$$

Assuming the same V_{DD}, the frequency of the TSVs in HBM is 1GHz compare to 18GHz for the LPRM. HBM however, utilizes both edges of the clock and therefore doubles the switching frequency. Despite the faster clocking speed of the TSVs in the LPMART, the reduction in the number of TSVs and the smaller capacitance cause the LPMART interface to consume approximately 40% less power than HBM (shown in Table 7). These power savings,

however, only deal with the TSVs in the DRAM stacks and do not include the power consumption of the DRAM cores. This is an area of ongoing research to provide definite results to backup these claims.

Table 7: Dynamic power estimate for HBM and LPMART.

	HBM	LPMART
$C \sim Length$	2.5	1
f_{sw}	2	18
Number of TSVs	1696	296
$\sim P_{dynamic}$	8480	5328

After calculating the power estimates, Ramulator simulations show the speed comparison between the LPMART scheme and other 3D stacked memories. Because the IE stations may have conflicts, where they cannot insert data onto the ring because other data already has that slot, the LPMART will have delayed latencies during busy bus periods. These latencies are minimal, however, since the activity of the data bus on a traditional HBM architecture is below 5% and a conflict will only delay data by one clock cycle (less than 0.06ns for an 18GHz clock). The smaller data bus of the LPMART, compared to HBM, causes decreased speeds but remains faster than the WideIO2 architecture. Unlike HBM, the LPMART will not have overhead between read and write commands because the ring is unidirectional. This reduces the column write latency and results in better performance for traces with more write requests. When looking at the performance results of the LPMART, it has speeds roughly in between WideIO2 and

HBM. With projected decreases in power consumption and manufacturing costs over HBM, the LPMART provides a new alternative to achieve faster speeds than WideIO2.

Figures 26 and 27 show the average latency incurred per command sent to the DRAM stack. As expected, HBM outperforms other 3D stacked memories in all traces. LPMART and WideIO2 follow a similar trend because both utilize the same DRAM dies. LPMART increases speeds over WideIO2 by 11%, but is slower than HBM by as much as 24%. For both the cache and no-cache configuration, the average latency per command maintained similar differences across all traces. The slower speeds of the LPMART and WideIO2 DRAM cores causes higher latencies per command, similar to the results shown for low power options in Figures 20 and 21.

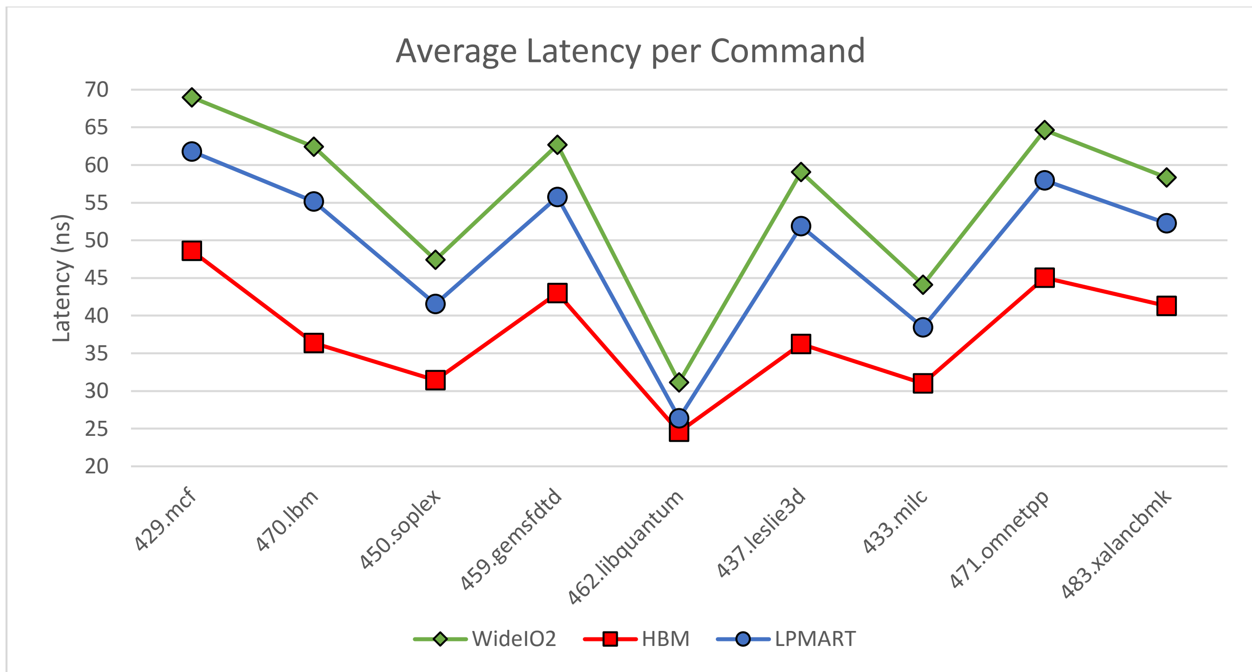

Figure 26: Average latency per command without cache (Appendix, Table 15).

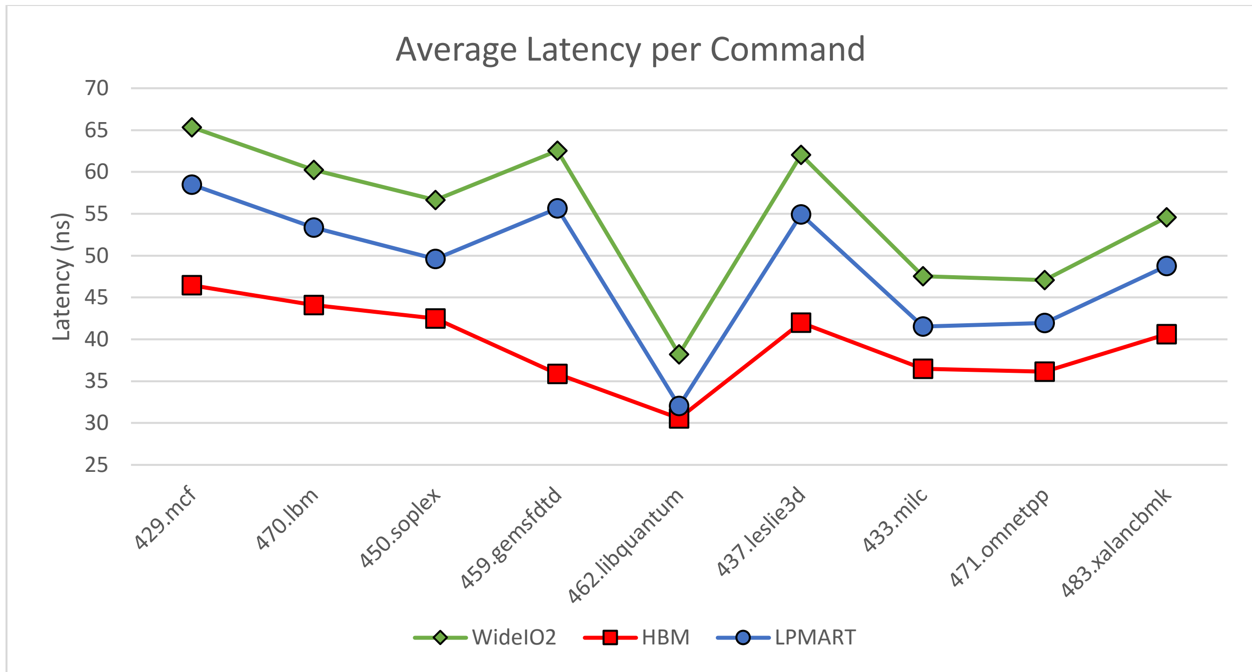

Figure 27: Average latency per command with cache (Appendix, Table 16).

To see a better comparison for the performance between the different 3D stacked memories, it is best to look at the total read latency. Here we see that, similar to the per command basis, HBM is the fastest in all cases while LPMART falls between HBM and WideIO2. LPMART is up to 17% faster than WideIO2, but can be up to 50% slower than HBM (Figure 28 and 29). Though this is a conservative range, LPMART is typically slower than HBM because of its narrow data bus. These statistics are similar for the cache configuration, but an overall reduction in latency occurs because of the reduction in DRAM requests. With a cache enabled, LPMART can be 5% slower than HBM and achieve speeds upwards of 20% faster than WideIO2.

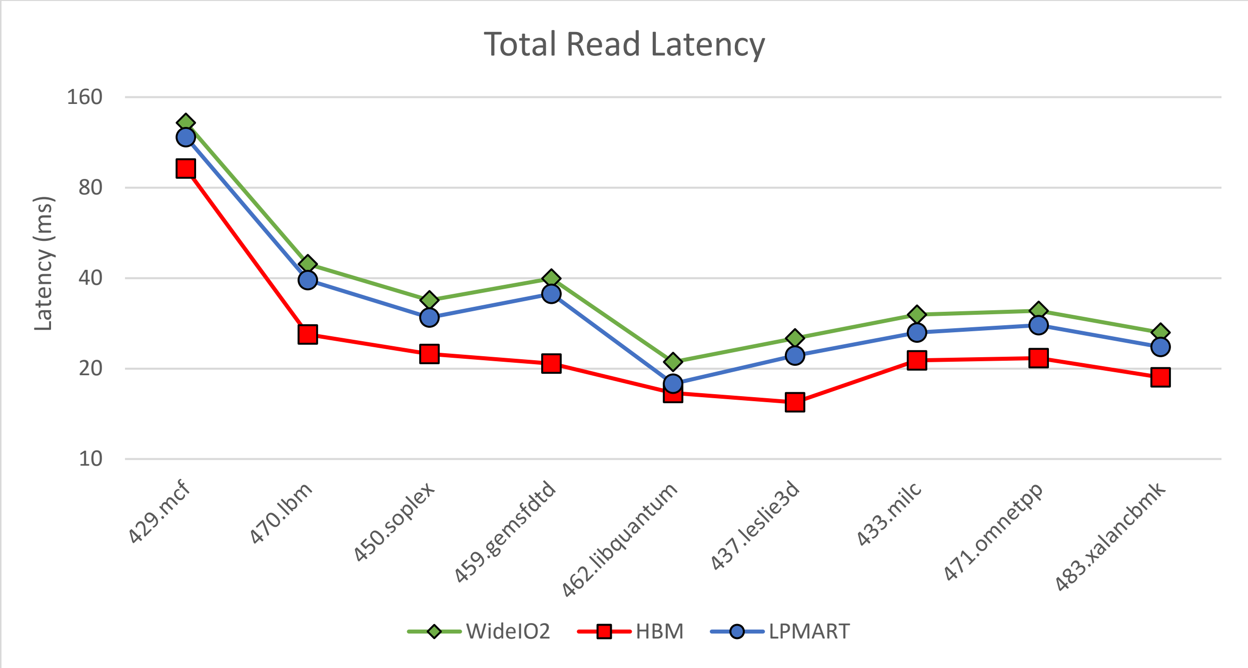

Figure 28: Total read latency without cache (Appendix, Table 17).

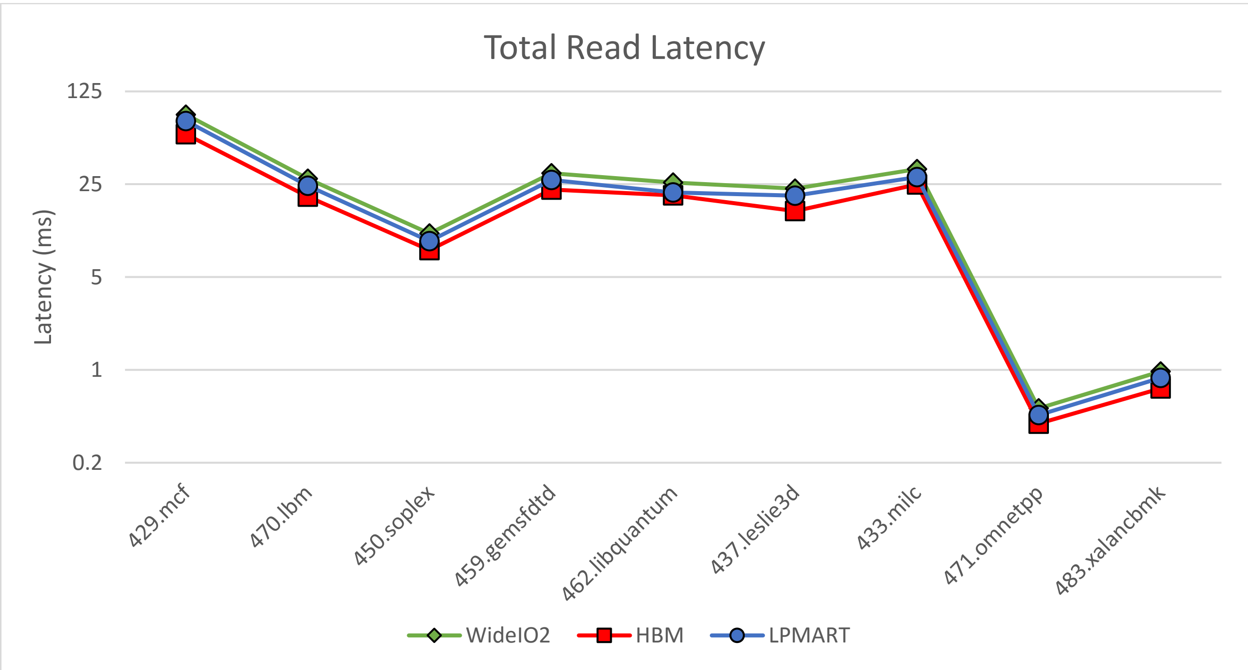

Figure 29: Total read latency with cache (Appendix, Table 18).

The final statistic measured between the 3D stacked memories is the average queue length for each channel. Because every 3D stacked memory used in these tests has 8 channels, the comparison shows the explicit speed seen by the communication scheme and the DRAM cores. With the cache disabled, the LPMART has the highest average queue length compared to WideIO2 and HBM (Figure 30). This is a result of the shared TSVs between all 8 channels and the conflicts that occur when the data bus is busy. Compared to WideIO2, LPMART has 23% larger queue lengths on average. When the cache is enabled, the difference seen between the 3D stacked memories is significantly reduced and the average queue lengths per channel are similar (Figure 31). Instead of a 23% increase, the queue length of the LPMART is only 4% larger than WideIO2.

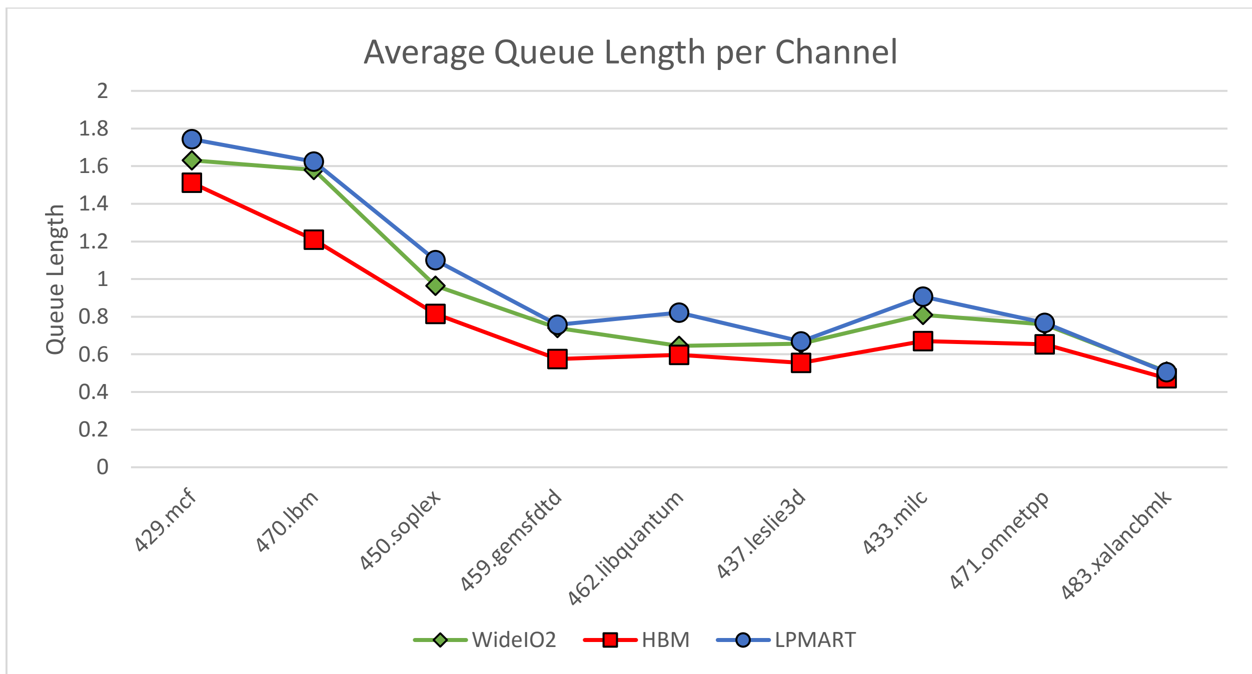

Figure 30: Average queue length without cache (Appendix, Table 19).

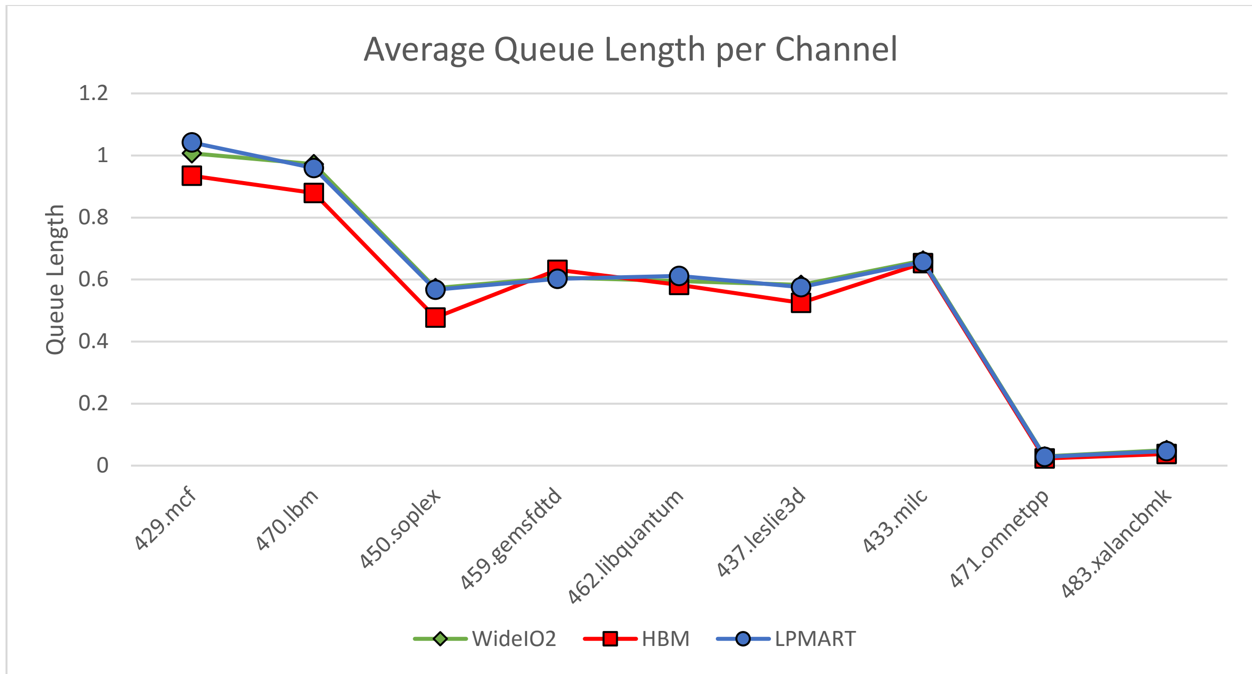

Figure 31: Average queue length with cache (Appendix, Table 20).

High Performance Ring Memories

By utilizing the ring in a 3D stacked memory, TSVs are reduced without compromising the performance. If the MART commination scheme is duplicated in Figure 16, the number of channels in the stack could be doubled from 8 to 16 (shown in Figure 32). With the data bus being 128-bits wide, the total number of TSVs used in a 16-channel architecture would be 848 (including address and control lines). Though this increases power consumption by approximately 80%, it still reduces manufacturing costs by utilizing fewer TSVs. With 16 channels, the parallelization for commands issued to the DRAM dies doubles and causes significant increases in speed over current HBM standards. By doubling the MART from 2 to 4 rings, four sets of 128-bit wide data busses can be used to create 32 independent channels. This would also double the power consumption but would only utilize 1696 TSVs, similar to HBM stacks. Table 8 shows the power estimates for the different channel architectures.

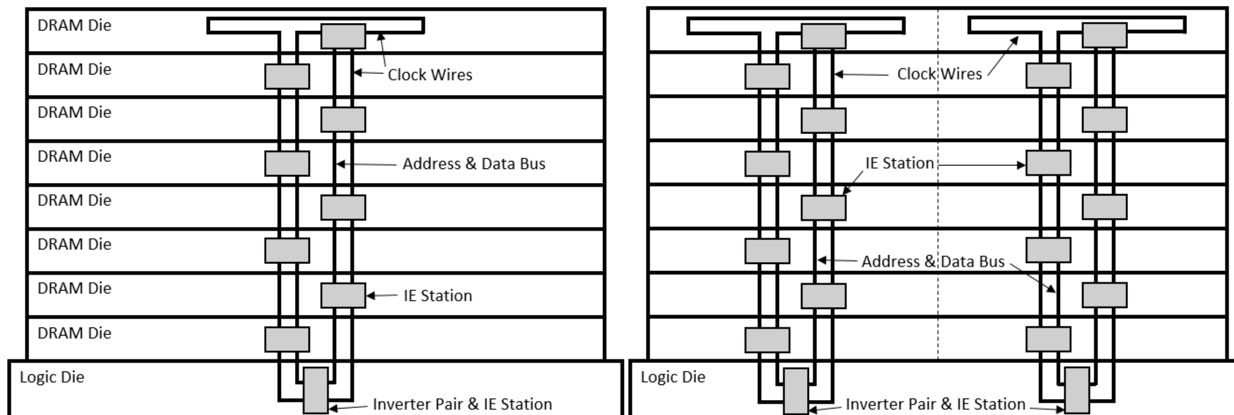

Figure 32: Side view of 8-channel (left) and 16-channel (right) MART for an 8GB stack.

Table 8: Dynamic power estimate for different channel counts.

	8-channel MART	16-channel MART	32-channel MART
Number of TSVs	424	848	1696
$\sim P_{dynamic}$	7632	15264	30528

By utilizing more channels, the DMC requests are evenly distributed across different sections of the DRAM dies. This reduces the time spent refreshing DRAM and allows concurrent commands to operate in parallel. To measure the performance for scalable channel counts, the MART was run in Ramulator with the same CPU traces. The average latency per command for the different channel counts saw few changes (Figure 33). These results show that more channels increase parallelization rather than single threaded commands. With the cache enabled, the latencies remained constant, but the difference between the different architectures decreased (Figure 34).

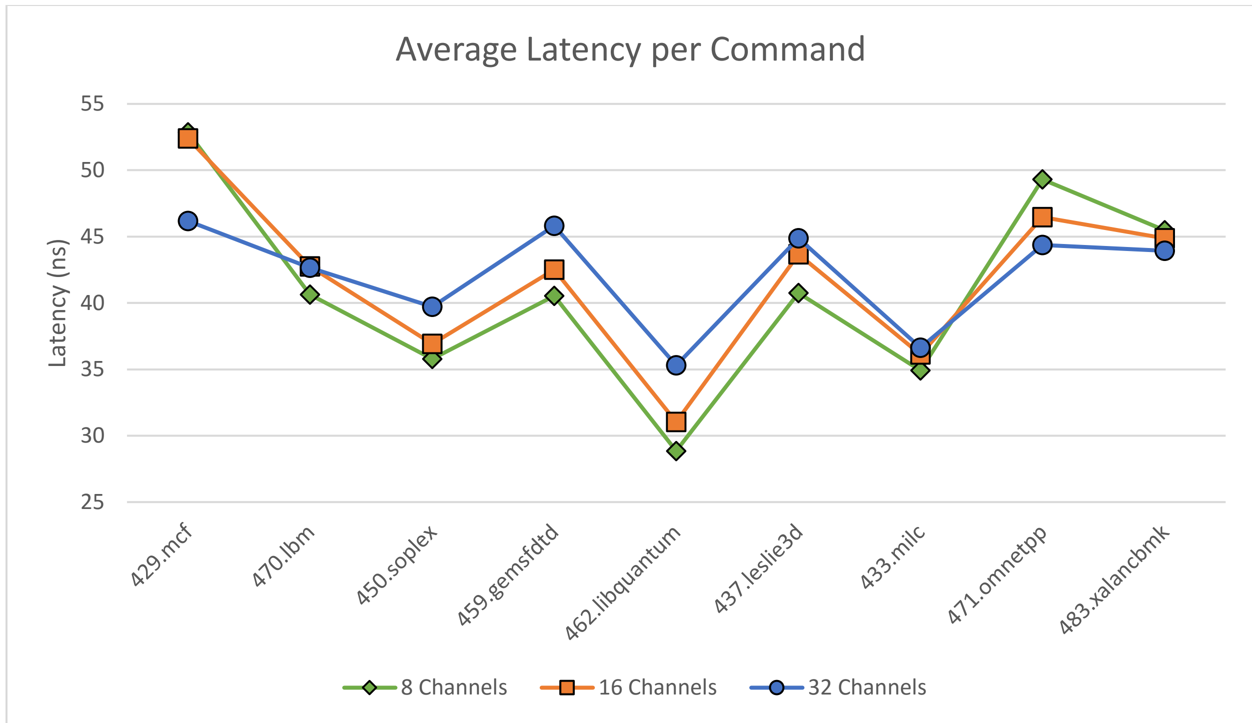

Figure 33: Average latency per command without cache (Appendix, Table 21).

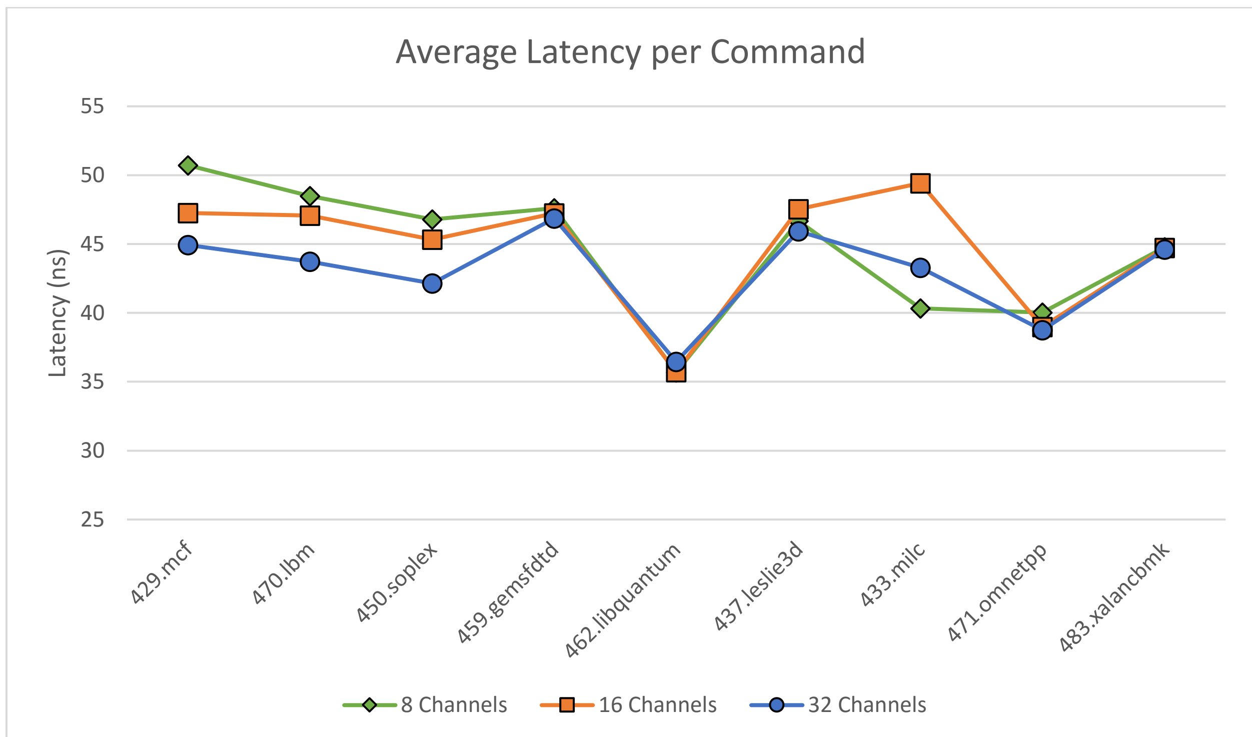

Figure 34: Average latency per command without cache (Appendix, Table 22).

Each time the channel count is doubled, the total read latency for all requests in a CPU trace is reduced by 100%. This is a result of increased concurrency in the DRAM dies and the reduction in the refresh time needed to ensure no data is lost. Figure 35 shows that for almost every CPU trace, the latency is cut in half. The exception comes from the 433.milc trace, which has a low cache catch rate, which in turn means that commands are not made to consecutive or similar locations in memory. This CPU trace also has a high write command percentage, which increases overhead in the internal DRAM cores. Figure 36 shows the same measurement with the cache enabled. Despite decreased latencies, doubling channel counts continues to double performance across most CPU traces.

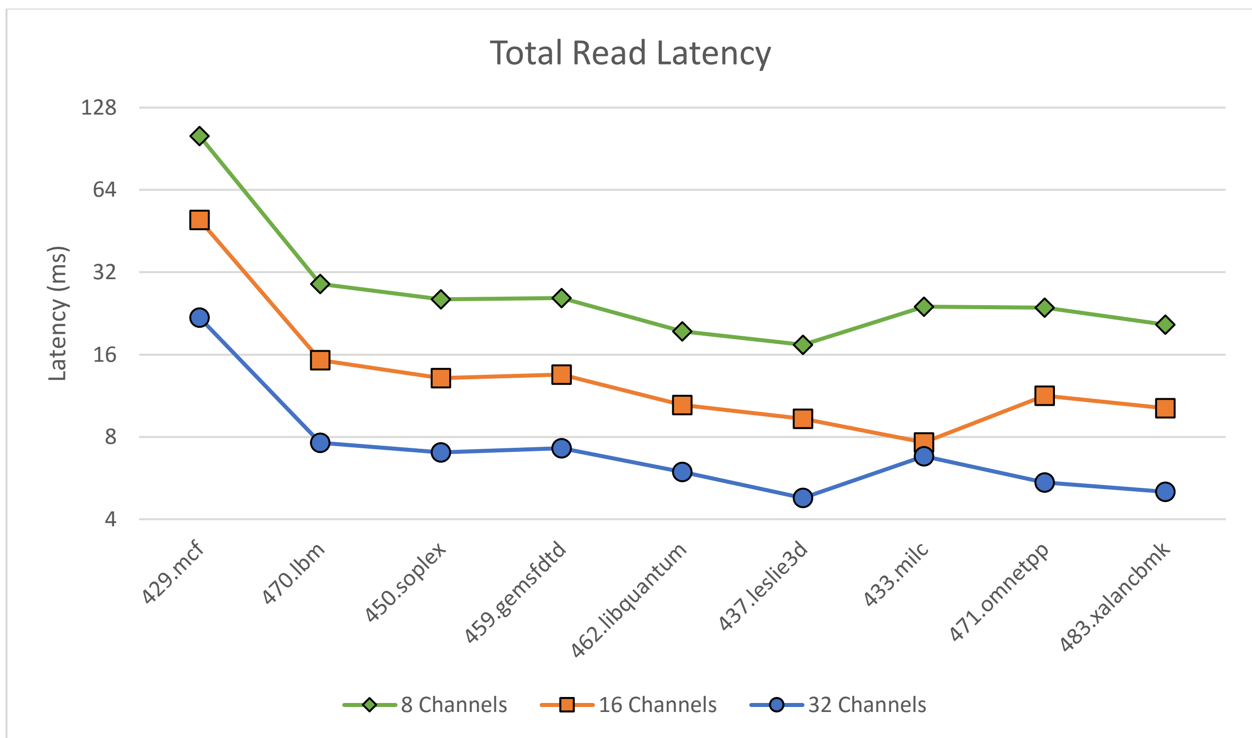

Figure 35: Total read latency without cache (Appendix, Table 23).

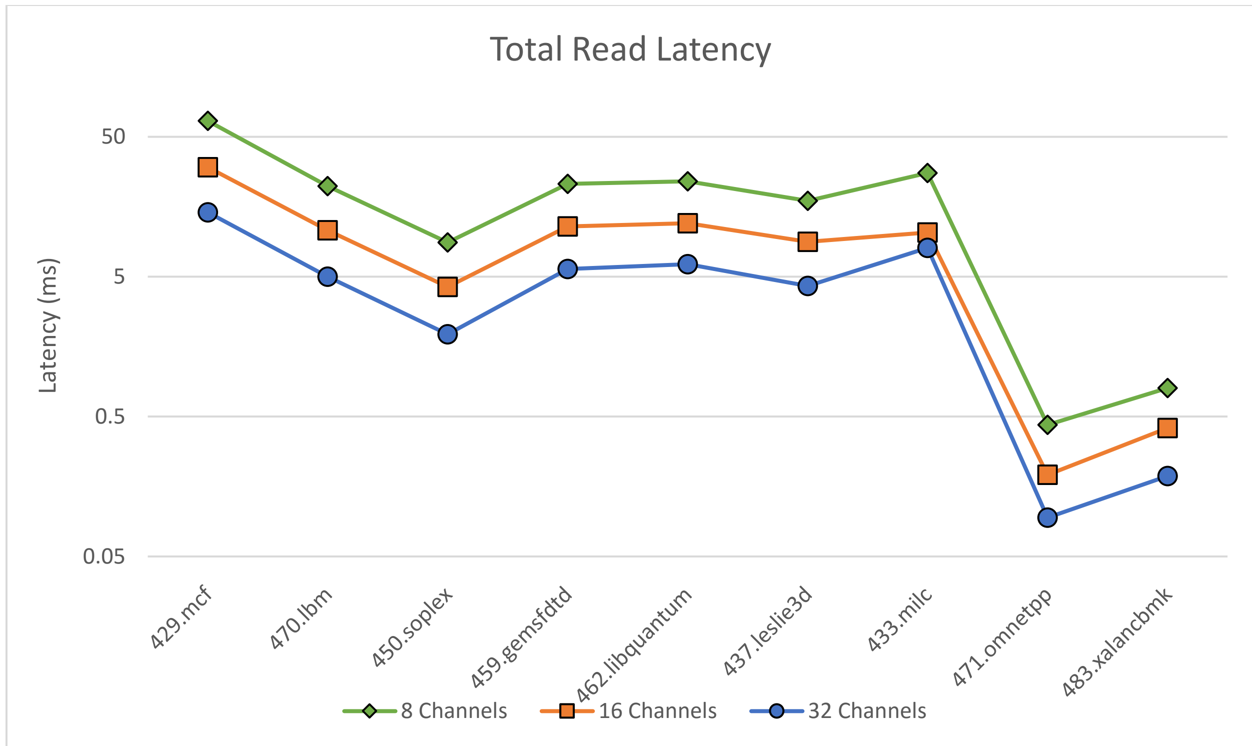

Figure 36: Total read latency with cache (Appendix, Table 24).

The final statistic measured was the average queue length for each channel configuration. It is important to note that these queue lengths are on a per channel basis; therefore, increasing the channel count divides the commands evenly amongst those channels. This reduces the total number of commands a channel encounters, which decreases congestion in the memory queues. There is a larger discrepancy between the 8-channel and the 16-channel configurations in terms of queue length. This decrease in the length of the queue is not as significant between the 16-channel and the 32-channel architectures. The reason for this is that a limit exists on the smallest the queue length can reach for each CPU trace. As the channel count increases, the queue length will approach values close to zero.

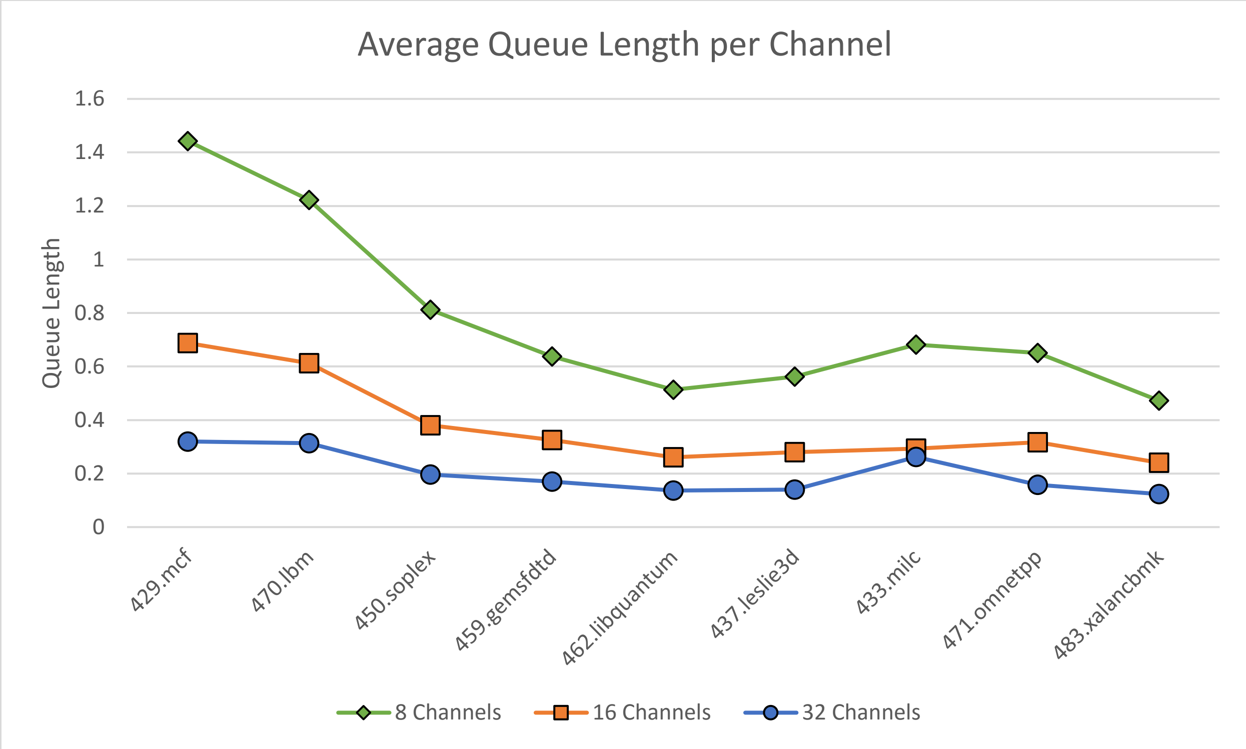

Figure 37: Average queue length without cache (Appendix, Table 25).

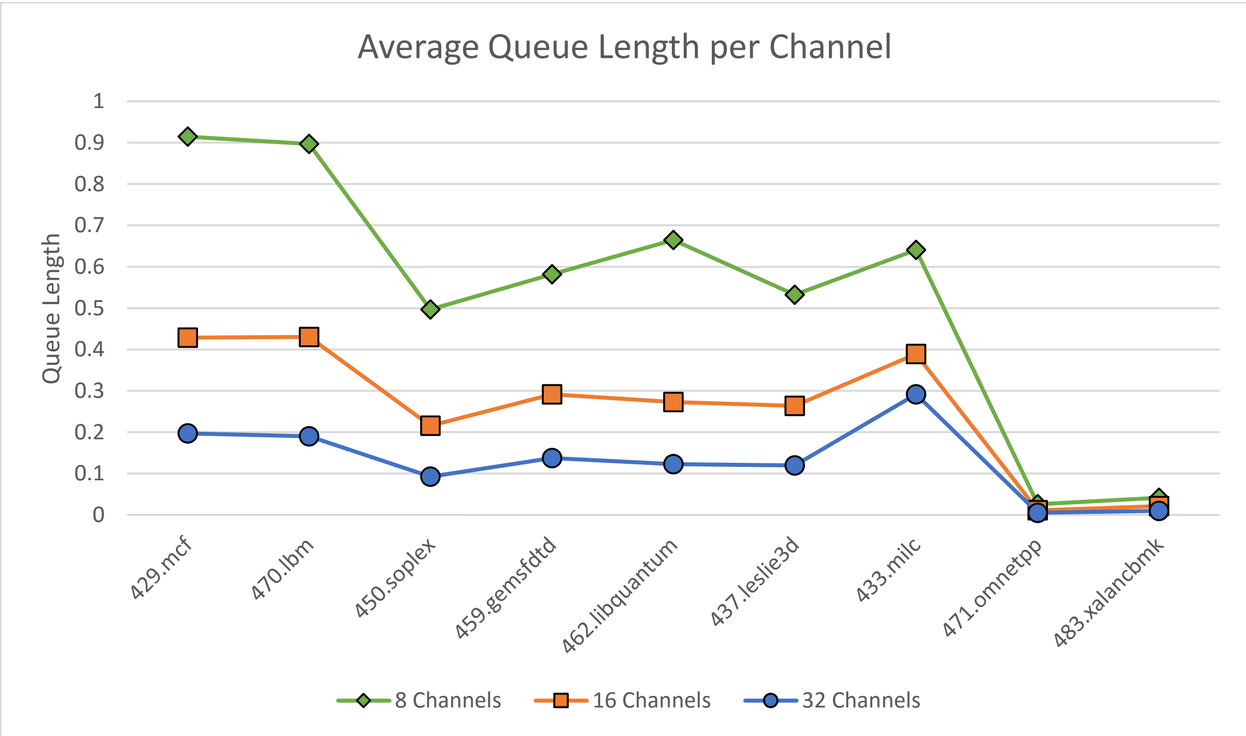

Figure 38: Average latency per command without cache (Appendix, Table 26).

CHAPTER IV

CONCLUSION

As memory continues to bottleneck system performance, 3D stacked memory can provide significant increases in throughput. With computer programs demanding more memory, fast data access is becoming necessary for high computing devices. These devices rely on large amounts of data for applications like virtual reality and DNA modelling. If these programs have quick access to their data, they can process more commands in a given period. This can have massive implications for fields like medical research that rely on data computations to study cells and symptoms of sickness.

Importance

By using a 3D resonant ring, the memory architecture using a ring-based topology (MART) can trade off power, throughput, and latency to match different application requirements. By using a narrow bus, and connecting it to all channels, the MART can provide low power consumption with speeds up to 20% faster than WideIO2. With multiple ring topologies in the same stack, the channel count can double from 8 to 16, and then to 32. This provides speeds up to 4 times faster than traditional HBM. This scalable architecture allows higher throughput and faster system performance for increasingly large programs. With all systems relying on memory to operate, the topology proposed in this paper provides significant impact on computing systems from lightweight IoT to large-scale data centers.

Future Research

The memory architecture using a ring-based topology provides a platform for future research on 3D memory communication schemes. By varying the number of 3D resonant rings,

the MART can accommodate different computer system requirements. To verify the low power options presented in this paper, ongoing research is occurring with simulation tools such as DRAMPower [34]. Utilizing HSPICE for the communication scheme and DRAMPower for the DRAM cores, researchers can verify the power consumption between different ring architectures, rather than utilizing the assumptions made in this paper. Research has also begun on verifying the operation of the IE stations, to ensure they can maintain the desired frequencies of the clock. Finally, research can look at new methods of incorporating the ring into other memory architectures. This includes 3D stacks based resistive random-access memory, such as 3D XPoint, and 3D SSDs. With more memory-based architectures becoming 3D, the ring-based architecture proposed in this paper can provide a fast and efficient communication scheme to improve system performance.

REFERENCES

[1] Jacob, Bruce, Spencer W. Ng, and David T. Wang. Memory Systems: Cache, DRAM, Disk. Burlington: Morgan Kaufmann Publishers, 2008. Print.

[2] Crisp, Richard. “Direct RAMbus technology: the new main memory standard.” *IEEE Micro* 17.6 (1997). Web.

[3] Przybylski, Steven. “Intel gambles on a sure DRAM thing.” *Electronic Engineering Times* 45.947 (1997). Web.

[4] “GDDR5 Part Catalog.” Micron Technology Inc., Feb. 2017. Web.

[5] Smith, Ryan. “The AMD Radeon R9 290 Review.” ANANDTECH, 5 Nov. 2013. Web.

[6] JEDEC Standard: GDDR5X SGRAM (JESD232A). JEDEC Solid State Technology Association. August 2016.

[7] Shilov, Anton. “GDDR5X Standard Finalized by JEDEC: New Graphics Memory up to 14 Gbps.” ANANDTECH, 22 Jan. 2016. Web.

[8] Parrish, Kevin. “The Sixth Generation of On-Board Memory For GPU Cards Won’t Arrive Until 2018.” Digital Trends, 22 Aug. 2016. Web.

[9] Chester, Edward. “Nvidia GTX Titan Black launched, set to be new performance king.” Bit-tech, 18 Feb. 2014. Web.

[10] Allan, Graham. “DDR4 Bank Groups in Embedded Applications.” *DesignWare Technical Bulletin*. Synopsys, 22 Dec. 2016. Web.

[11] JEDEC Standard: DDR4 SDRAM (JESD79-4). JEDEC Solid State Technology Association. September 2012.

[12] JEDEC Standard: DOUBLE DATA RATE SDRAM (JESD79F). JEDEC Solid State Technology Association. February 2008.

[13] JEDEC Standard: DDR2 SDRAM (JESD79-2F). JEDEC Solid State Technology Association. November 2009.

[14] JEDEC Standard: DDR3 SDRAM (JESD79-3F). JEDEC Solid State Technology Association. July 2012.

[15] JEDEC Standard No. 21-C. Page 3.11.5.2. JEDEC Solid State Technology Association.

[16] Nair, Prashant, Chia-Chen Chou, and Moinuddin Qureshi. “A Case for Refresh Pausing in DRAM Memory Systems.” *IEEE High Performance Computer Architecture* 19 (2013): 627-38. Web.

[17] “High-Bandwidth Memory, Reinventing Memory Technology.” Advanced Micro Devices, Inc., 2013. Web.

[18] Wasson, Scott. “AMD’s high-bandwidth memory explained.” *The Tech Report: PC Hardware Explored.* The Tech Report, 19 May 2015. Web.

[19] JEDEC Standard: HBM (JESD235A). JEDEC Solid State Technology Association. November 2015.

[20] “HBM2 Deep Dive.” Monitor Insider, 2 Feb. 2016. Web.

[21] Jun, Hongshin, Dr. “HBM (High Bandwidth Memory) for 2.5D.” *Semicon Taiwan 2015.* SK Hynix Inc., Sept. 2015. Web.

[22] Shilov, Anton. “JEDEC Publishes HBM2 Specification as Samsung Begins Mass Production of Chips.” ANANDTECH, 20 Jan. 2016. Web.

[23] Walton, Mark. “HBM3: Cheaper, up to 64GB on-package, and terabytes-per-second bandwidth.” ARS Technica, 23 Aug. 2016. Web.

[24] JEDEC Standard: WIDE I/O SINGLE DATA RATE (JESD229). JEDEC Solid State Technology Association. December 2011.

[25] JEDEC Standard: WIDE I/O 2 (JESD229-2). JEDEC Solid State Technology Association. August 2014.

[26] Ahn, K., and C. Yoo. “Skew cancellation technique for >256-Gbyte/s high-bandwidth memory (HBM).” *Electronics Letters* 52.13 (2016): 1155-157. Web.

[27] Thakkar, Ishan, and Sudeep Pasricha. “Massed Refresh: AN Energy-Efficient Technique to Reduce Refresh Overhead in Hybrid Memory Cube Architectures.” *IEEE* (2016): n. pag. Web.

[28] Lee, Donghyuk, Gennady Pekhimenko, Samira Khan, Saugata Chose, and Onur Mutlu. “Simultaneous Multi Layer Access: A High Bandwidth and Low Cost 3D-Stacked Memory Interface.” *SAFARI Technical Report* 2015.008 (2015). Web.

[29] Mandal, Ayan. *Efficient Design and Clocking for a Network-on-Chip*. Diss. Texas A&M University, 2013. Web. 8 Aug. 2016.

[30] Cordero, Victor. *Clock Distribution Scheme using Coplanar Transmission Lines*. Diss. Texas A&M University, 2008. Web. 12 Aug. 2016.

[31] Kim, Yoongu, Weikun Yan, and Onur Mutlu. “Ramulator: A Fast and Extensible DRAM Simulator.” *IEEE Computer Architecture Letters* 15.1 (2015): 45-49. Web.

[32] Cao, Yu. “Latest Models.” *Predictive Technology Model*. Nanoscale Integration and Modeling (NIMO) Group, ASU., 30 Sept. 2008. Web.

[33] Denneman, Frank. “Memory Deep Dive: Memory Subsystem Organization.” *Frankdenneman.nl.* WordPress, 18 Feb. 2015. Web.

[34] Chandrasekar, Karthik, Christian Weis, Yonghui Li, Sven Foossens, Matthias Jung, Omar Naji, Benny Akesson, Norbert When, and Kees Goossens. DRAMPower: Open-source DRAM Power & Energy Estimation Tool. Web.

APPENDIX

MEMORY SIMULATION DATA TABLES

Current DRAM Speeds

Table 9: Average latency per command without cache in nanoseconds.

CPU Trace	LPDDR4	LPDDR3	DDR3	DDR4	GDDR5	WideIO2	HBM
429.mcf	87.5	86.5	51.2	43.6	48.9	69.0	48.6
470.lbm	69.1	62.7	31.3	26.2	39.6	62.4	36.3
450.soplex	56.7	50.4	28.6	24.7	30.1	47.4	31.4
459.gemsfdtd	58.3	54.7	27.8	25.4	38.2	62.7	43.0
462.libquantum	36.5	30.2	20.0	19.8	20.3	31.1	24.6
437.leslie3d	54.2	49.0	28.2	27.3	36.4	59.1	36.2
433.milc	52.5	46.8	28.3	26.9	29.1	44.1	31.0
471.omnetpp	69.8	67.7	40.0	39.5	44.1	64.7	45.0
483.xalancbmk	56.3	53.0	31.5	33.3	38.8	58.3	41.3

Note: See page 32.

Table 10: Average latency per command with cache in nanoseconds.

CPU Trace	LPDDR4	LPDDR3	DDR3	DDR4	GDDR5	WideIO2	HBM
429.mcf	80.7	79.5	45.1	40.8	47.7	65.3	46.5
470.lbm	73.1	68.7	34.3	27.4	42.6	60.3	44.1
450.soplex	76.7	70.5	36.9	29.7	39.7	56.6	42.5
459.gemsfdtd	63.0	59.0	29.8	28.0	42.9	62.5	35.9
462.libquantum	47.1	42.2	25.7	23.2	27.1	38.2	30.6
437.leslie3d	66.3	60.8	32.6	29.0	42.4	62.0	42.0
433.milc	62.4	56.9	32.9	28.6	34.1	47.5	36.5
471.omnetpp	50.9	48.0	29.6	31.2	36.8	47.1	36.1
483.xalancbmk	60.0	56.9	34.5	33.4	43.1	54.6	40.6

Note: See page 33.

Table 11: Total read latency without cache in milliseconds.

CPU Trace	LPDDR4	LPDDR3	DDR3	DDR4	GDDR5	WideIO2	HBM
429.mcf	667	659	390	166	143	132	92.7
470.lbm	197	179	89.4	37.3	42.7	44.7	25.9
450.soplex	161	143	81.3	35.2	32.3	33.8	22.4
459.gemsfdtd	148	139	70.7	32.2	36.3	39.8	20.8
462.libquantum	98.5	81.5	54.1	26.8	20.4	21.1	16.6
437.leslie3d	92.6	86.7	48.2	23.3	23.2	25.2	15.4
433.milc	63.3	56.4	34.1	26.9	24.7	30.3	21.3
471.omnetpp	136	132	77.8	38.8	32.8	31.1	21.7
483.xalancbmk	104	97.7	58.2	30.8	26.9	26.4	18.7

Note: See page 34.

Table 12: Total read latency with cache in milliseconds.

CPU Trace	LPDDR4	LPDDR3	DDR3	DDR4	GDDR5	WideIO2	HBM
429.mcf	413	407	232	104	93.5	83.7	59.5
470.lbm	134	126	62.7	25.1	29.9	27.5	20.1
450.soplex	57.8	53.1	27.8	11.2	11.6	10.6	7.98
459.gemsfdtd	121.9	114.2	57.6	27.1	31.6	30.3	22.8
462.libquantum	127	114	69.3	31.3	28.1	25.8	20.6
437.leslie3d	98.9	90.8	48.7	21.7	24.0	23.2	15.7
433.milc	74.0	67.6	39.0	28.4	28.9	32.4	24.9
471.omnetpp	2.16	2.07	1.28	0.76	0.64	0.51	0.39
483.xalancbmk	4.56	4.27	2.59	1.24	1.19	0.97	0.73

Note: See page 34.

Table 13: Average queue length per channel without cache.

CPU Trace	LPDDR4	LPDDR3	DDR3	DDR4	GDDR5	WideIO2	HBM
429.mcf	7.18	7.15	7.75	3.57	2.48	1.63	1.51
470.lbm	7.88	8.18	5.89	2.19	2.12	1.58	1.21
450.soplex	4.91	5.06	4.95	1.85	1.47	0.96	0.82
459.gemsfdtd	3.14	3.29	2.72	1.20	1.02	0.74	0.57
462.libquantum	3.63	3.59	3.41	1.23	1.03	0.65	0.59
437.leslie3d	3.19	3.26	2.52	1.05	0.92	0.66	0.56
433.milc	1.83	1.83	2.06	1.12	0.98	0.81	0.67
471.omnetpp	3.57	3.56	3.24	1.34	1.06	0.76	0.65
483.xalancbmk	2.07	2.02	1.90	0.91	0.71	0.51	0.47

Note: See page 35.

Table 14: Average queue length per channel with cache.

CPU Trace	LPDDR4	LPDDR3	DDR3	DDR4	GDDR5	WideIO2	HBM
429.mcf	5.01	5.09	4.84	1.72	1.43	1.01	0.93
470.lbm	5.65	5.73	4.30	1.19	1.30	0.97	0.88
450.soplex	3.60	3.65	2.46	0.69	0.80	0.57	0.48
459.gemsfdtd	2.88	2.93	2.16	0.84	0.85	0.61	0.63
462.libquantum	3.92	4.02	2.97	1.02	1.11	0.59	0.58
437.leslie3d	4.05	4.07	2.91	0.78	0.81	0.58	0.52
433.milc	1.71	1.69	1.45	0.76	0.75	0.66	0.65
471.omnetpp	0.18	0.17	0.11	0.04	0.04	0.03	0.02
483.xalancbmk	0.33	0.31	0.19	0.07	0.07	0.05	0.04

Note: See page 36.

Low Power Ring Memory

Table 15: Average latency per command without cache in nanoseconds.

CPU Trace	WideIO2	HBM	LPRM
429.mcf	69.0	48.6	61.8
470.lbm	62.4	36.3	55.1
450.soplex	47.4	31.4	41.5
459.gemsfdtd	62.7	43.0	55.8
462.libquantum	31.1	24.6	26.4
437.leslie3d	59.1	36.2	51.9
433.milc	44.1	31.0	38.4
471.omnetpp	64.7	45.0	57.9
483.xalancbmk	58.3	41.3	52.2

Note: See page 39.

Table 16: Average latency per command with cache in nanoseconds.

CPU Trace	WideIO2	HBM	LPRM
429.mcf	65.3	46.5	58.5
470.lbm	60.3	44.1	53.3
450.soplex	56.6	42.5	49.6
459.gemsfdtd	62.5	35.9	55.6
462.libquantum	38.2	30.6	32.1
437.leslie3d	62.0	42.0	54.9
433.milc	47.5	36.5	41.5
471.omnetpp	47.1	36.1	41.9
483.xalancbmk	54.6	40.6	48.8

Note: See page 40.

Table 17: Total read latency without cache in milliseconds.

CPU Trace	WideIO2	HBM	LPRM
429.mcf	132	92.7	118
470.lbm	44.7	25.9	39.4
450.soplex	33.8	22.4	29.6
459.gemsfdtd	39.8	20.8	35.4
462.libquantum	21.1	16.6	17.8
437.leslie3d	25.2	15.4	22.1
433.milc	30.3	21.3	26.4
471.omnetpp	31.1	21.7	27.9
483.xalancbmk	26.4	18.7	23.6

Note: See page 41.

Table 18: Total read latency with cache in milliseconds.

CPU Trace	WideIO2	HBM	LPRM
429.mcf	83.7	59.5	74.9
470.lbm	27.5	20.1	24.4
450.soplex	10.6	7.98	9.33
459.gemsfdtd	30.3	22.8	26.9
462.libquantum	25.8	20.6	21.6
437.leslie3d	23.2	15.7	20.5
433.milc	32.4	24.9	28.4
471.omnetpp	0.51	0.39	0.46
483.xalancbmk	0.97	0.73	0.87

Note: See page 41.

Table 19: Average queue length per channel without cache.

CPU Trace	WideIO2	HBM	LPRM
429.mcf	1.63	1.51	1.74
470.lbm	1.58	1.21	1.62
450.soplex	0.96	0.82	1.10
459.gemsfdtd	0.74	0.57	0.76
462.libquantum	0.65	0.59	0.82
437.leslie3d	0.66	0.56	0.67
433.milc	0.81	0.67	0.91
471.omnetpp	0.76	0.65	0.77
483.xalancbmk	0.51	0.47	0.51

Note: See page 42.

Table 20: Average queue length per channel with cache.

CPU Trace	WideIO2	HBM	LPRM
429.mcf	1.01	0.93	1.04
470.lbm	0.97	0.88	0.96
450.soplex	0.57	0.48	0.57
459.gemsfdtd	0.61	0.63	0.61
462.libquantum	0.59	0.58	0.61
437.leslie3d	0.58	0.52	0.58
433.milc	0.66	0.65	0.66
471.omnetpp	0.03	0.02	0.03
483.xalancbmk	0.05	0.04	0.05

Note: See page 43.

High Performance Ring Memory

Table 21: Average latency per command without cache in nanoseconds.

CPU Trace	8 Channels	16 Channels	32 Channels
429.mcf	52.9	52.4	46.2
470.lbm	40.6	42.8	42.7
450.soplex	35.8	36.9	39.7
459.gemsfdtd	40.5	42.5	45.8
462.libquantum	28.9	31.0	35.3
437.leslie3d	40.8	43.7	44.9
433.milc	34.9	36.1	36.6
471.omnetpp	49.3	46.5	44.4
483.xalancbmk	45.5	44.9	43.9

Note: See page 45.

Table 22: Average latency per command with cache in nanoseconds.

CPU Trace	8 Channels	16 Channels	32 Channels
429.mcf	50.7	47.3	44.9
470.lbm	48.5	47.1	43.7
450.soplex	46.8	45.3	42.1
459.gemsfdtd	47.6	47.2	46.9
462.libquantum	35.7	35.7	36.5
437.leslie3d	46.7	47.5	45.9
433.milc	40.3	49.4	43.3
471.omnetpp	40.0	39.0	38.8
483.xalancbmk	44.8	44.7	44.6

Note: See page 45.

Table 23: Total read latency without cache in milliseconds.

CPU Trace	8 Channels	16 Channels	32 Channels
429.mcf	100.8	49.7	21.8
470.lbm	29.0	15.3	7.61
450.soplex	25.5	13.1	7.01
459.gemsfdtd	25.7	13.5	7.28
462.libquantum	19.5	10.5	5.96
437.leslie3d	17.4	9.31	4.79
433.milc	23.9	7.64	6.80
471.omnetpp	23.7	11.3	5.45
483.xalancbmk	20.6	10.2	5.05

Note: See page 46.

Table 24: Total read latency with cache in milliseconds.

CPU Trace	8 Channels	16 Channels	32 Channels
429.mcf	65.0	30.3	14.4
470.lbm	22.2	10.8	5.0
450.soplex	8.80	4.23	1.94
459.gemsfdtd	23.0	11.4	5.67
462.libquantum	24.1	12.1	6.15
437.leslie3d	17.4	8.87	4.29
433.milc	27.5	10.4	8.03
471.omnetpp	0.44	0.19	0.09
483.xalancbmk	0.80	0.41	0.19

Note: See page 47.

Table 25: Average queue length per channel without cache.

CPU Trace	8 Channels	16 Channels	32 Channels
429.mcf	1.44	0.69	0.32
470.lbm	1.22	0.61	0.31
450.soplex	0.81	0.38	0.19
459.gemsfdtd	0.64	0.33	0.17
462.libquantum	0.51	0.26	0.14
437.leslie3d	0.56	0.28	0.14
433.milc	0.68	0.29	0.26
471.omnetpp	0.65	0.32	0.16
483.xalancbmk	0.47	0.24	0.12

Note: See page 48.

Table 26: Average queue length per channel with cache.

CPU Trace	8 Channels	16 Channels	32 Channels
429.mcf	0.92	0.43	0.20
470.lbm	0.89	0.43	0.19
450.soplex	0.50	0.22	0.09
459.gemsfdtd	0.58	0.29	0.14
462.libquantum	0.67	0.27	0.12
437.leslie3d	0.53	0.26	0.12
433.milc	0.64	0.39	0.29
471.omnetpp	0.03	0.01	0.01
483.xalancbmk	0.04	0.02	0.01

Note: See page 48.