UNIVERSITY OF CALIFORNIA, SAN DIEGO

Onyx: A Phase-Change Memory Storage Array

A Thesis submitted in partial satisfaction of the requirements
for the degree Master of Science

in

Computer Science

by

Ameen Daniel Akel

Committee in charge:

Professor Steven Swanson, Chair
Professor Allan Snavely
Professor Michael Taylor

2011

The Thesis of Ameen Daniel Akel is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

Chair

University of California, San Diego

2011

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF GRAPHS

LIST OF TABLES

ACKNOWLEDGEMENTS

I am grateful for the opportunity presented to me by my advisor–Dr. Steven Swanson–to conduct meaningful research as a member of the Non-Volatile Systems Laboratory (NVSL). Without his trust, guidance, and support, I could not have completed this thesis.

I would also like to thank my colleagues at the NVSL: Adrian Caulfield, Laura Grupp, Joel Coburn, Todor Mollov, Hung-Wei Tseng, Arup De, Michael Wei, Alex Eisner, Trevor Bunker, Sundaram Bhaskaran, and Pravin Prabhu.

Of my colleagues, special thanks goes to Adrian Caulfield and Todor Mollov for their help with many aspects of my thesis: the PCM DIMM design, their suggestions for the Onyx memory controller design, and for their patience and help as I learned about the Moneta system.

Of my colleagues and professors at both the NVSL and SDSC, my thanks goes to the authors of the papers of which I based Onyx: Adrian Caulfield, Todor Mollov, Arup De, Joel Coburn, Jiahua He, Arun Jagatheesan, Dr. Steven Swanson, Dr. Allan Snavely, and Dr. Rajesh Gupta.

Special mention goes to my parents: Eli and Mireille Akel for their support (both monetary and otherwise) throughout my academic career. Without their aid, I would not have had the opportunity to attend college or graduate school.

Lastly, I offer my thanks to my many friends (both in San Diego and abroad) who supported me throughout my academic career.

Portions of the content in Chapter 4 appeared in the Proceedings of the 3rd USENIX Conference on Hot Topics in Storage and File Systems, 2011. A. Akel, A. M. Caulfield, T. I. Mollov, R. K. Gupta, and S. Swanson. Onyx: A Prototype Phase Change Memory Storage Array. The thesis author was the primary investigator and author of this paper.

Portions of the content in Chapter 5 appeared in the Proceedings of the 3rd USENIX Conference on Hot Topics in Storage and File Systems, 2011. A. Akel, A. M. Caulfield, T. I. Mollov, R. K. Gupta, and S. Swanson. Onyx: A Prototype Phase Change Memory Storage Array. The thesis author was the primary investigator and author of this paper.

ABSTRACT OF THE THESIS

Onyx: A Phase-Change Memory Storage Array

by

Ameen Daniel Akel

Master of Science in Computer Science

University of California, San Diego, 2011

Professor Steven Swanson, Chair

This thesis describes a prototype, high-performance, solid-state drive based on first-generation phase-change memory (PCM) devices called *Onyx*. Onyx has a capacity of 10 GB and connects to a host machine via PCI-Express. This thesis also describes the PCM DIMMs that make up Onyx, including the PCM memory devices used, and the FPGA-based controller that manages the PCM DIMMs. This thesis provides insight into the PCM DIMM design process and shows the changes required to integrate future-generation PCM. This thesis also describes my experience with an existing wear-leveling technique, which I implemented in Onyx in order to manage wear. My results show that Onyx outperforms a state-of-the-art FusionIO ioDrive flash memory SSD for both reads and small writes. Also, I show that Onyx also incurs significantly less CPU overhead per IOP for

small requests, which has the advantage of saving power. My results also show that a first-generation PCM-based SSD performs on par with a flash-based SSD in real-world workloads. As part of my study, I address the performance impact of start-gap wear-leveling in Onyx.

Chapter 1

Introduction

Flash memory has recently been the front-runner in the revolution of fast, non-volatile storage systems. Bandwidth-hungry data consumers have found success with fast, flash-memory-based systems such as FusionIO's ioDrive [8], among other offerings. While flash memory provides significant performance increases over other non-volatile technologies (such as disk and tape), it does so with a significant increase in complexity. Flash Translation Layers (or FTLs) add both complexity and latency to flash-based systems, which limits the performance of these new flash-based solid-state drives (SSDs). A new non-volatile memory technology–phase-change memory (PCM)–is poised to provide similar or better gains than flash memory, while avoiding the increased complexity brought by FTLs in flash-based SSDs.

Predictions for future generations of PCM show that it will scale better than flash memory in terms of density and performance [9]. PCM also allows for in-place updates of data, versus the cumbersome erase and program process of flash memory. This advantage also removes the necessity for a complex translation layer (or FTL for flash memory), which decreases overall request latency for small requests, lower total system power, and reduce overall CPU usage in a PCM-based storage system. However, PCM exhibits its own usage quirks: non-uniform read vs. write access times, wearout challenges, among other issues, which I explore in this study.

In this work, I have built *Onyx* [3]: A system based on the Moneta SSD

with PCM DIMMs in place of the original DDR2 DRAM. Onyx is a PCI-Express-attached PCM SSD controlled by a highly optimized block driver. Onyx contains 8 GB of logically addressable PCM and an additional 2 GB of "out-of-band" PCM storage. In this paper, I describe the design of the PCM DIMMs that comprise Onyx and Onyx's PCM controller.

I also present results which compare the performance of Onyx to a state-of-the-art flash-based, PCI-Express-attached SSD from FusionIO and the original Moneta hardware. I found that Onyx performs at over 1.1 GB/s for reads and outperforms datasheet expectations by up to 34% for writes. The results also show that Onyx outperforms the FusionIO flash SSD for reads. However, although Onyx outperforms the flash SSD for small writes under 2 KB, the flash SSD outperforms Onyx for larger write requests.

The key observation from Onyx, and its comparison to the flash-based SSD, is that correctly-architected PCM-based storage arrays have become and will remain formidable competitors for high-end flash-based SSDs, especially for workloads which require frequent small-sized accesses or workloads dominated by read requests.

Chapter 2

Previous Work

2.1 Moneta SSD

Much of the Onyx SSD is based upon from the Moneta SSD design [7] [6]. Moneta is a fast, PCI-Express attached SSD. Moneta is comprised of both a hardware and software stack that aim to minimize latency and maximize concurrency. This section describes the hardware and software of Moneta.

Hardware Description

Moneta (shown in Figure 2.1) contains a central control unit called the *brain*, a ring network, and eight memory controllers. The Moneta brain controls and tracks requests as they move throughout the SSD: The brain receives requests, schedules those requests throughout the system, and stores request information in a scoreboard. The brain also includes a set of transfer buffers which store the data it receives or sends via DMA from or to the host machine. The Moneta brain interfaces with the rest of the system via a ring network. The ring network also connects the eight memory controllers, which interface with the memory in the DIMM slots.

The Moneta system (whose architecture is shown in Figure 2.2) is built atop the BEE3 system [5] developed for use in the RAMP project [15]. Moneta connects to a host machine via an eight-lane PCI-Express 1.1 connection, and can handle

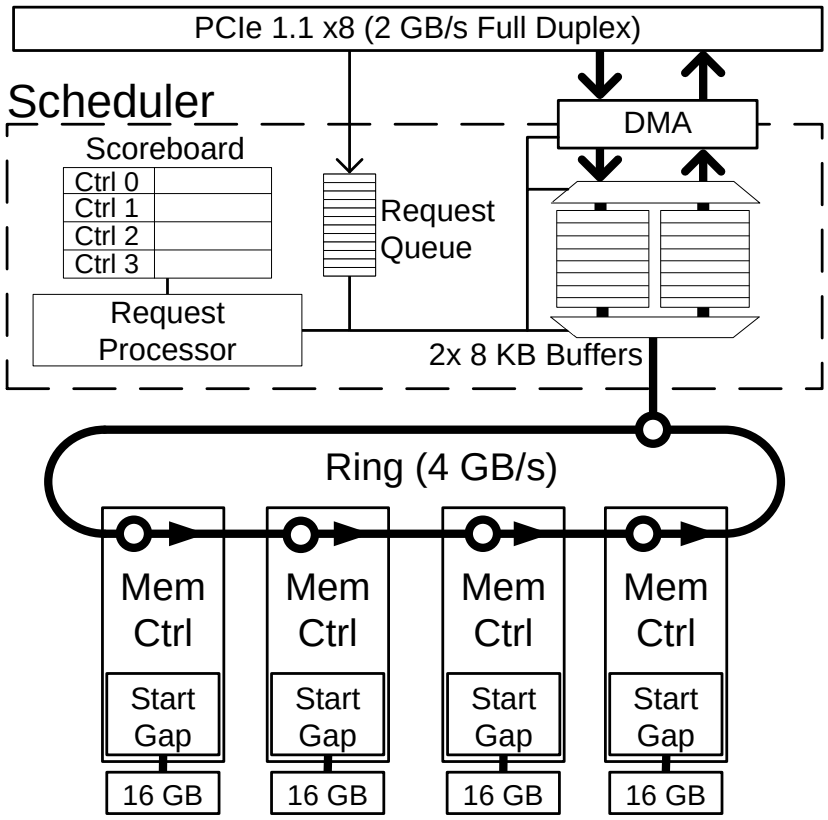


Figure 2.1: This shows the original version of Moneta's high-level architecture.

up to 64 oustanding requests at a time. Four FPGAs connected in a ring–each connected to two banks of two DDR2 DIMM slots–make up the physical Moneta hardware.

Software Description

Moneta's software stack consists of the operating system IO stack (Linux, in Moneta) and the Moneta kernel driver. The driver issues requests through PCI-Express to the Moneta hardware. The Moneta kernel driver is optimized for parallel, low-latency accesses to hardware. It bypasses the Linux IO request scheduler, utilizes atomic PIO writes to issue requests and limit contention, processes completed tags in parallel, and takes advantage of spin-waits to decrease latency.

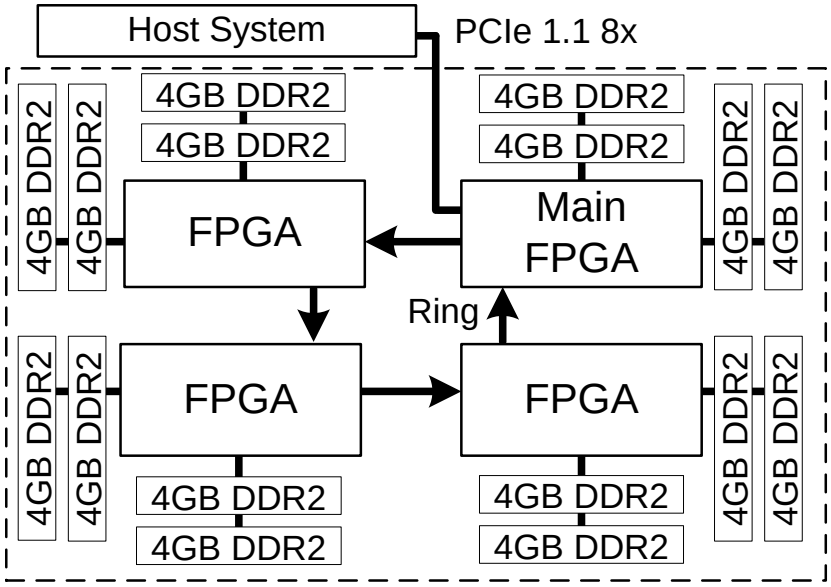


Figure 2.2: This shows the BEE3 prototyping system architecture. Both Moneta and Onyx were built atop the BEE3 platform.

2.2 Start-Gap Wear-Leveling

Onyx employs a low-overhead wear-leveling technique developed at IBM Research called start-gap wear-leveling [14]. The goal of start-gap is to distribute the wear of a non-uniform write access pattern uniformly across the memory array, such that the technique limits the wear on a single location in memory. Start-gap does this by rotating the logical to physical address mapping for an array after a fixed number of writes to that array. The technique accomplishes this with only two registers: a *start* register and a *gap* register each representing a logical *line* in memory, and a few gates worth of comparison and arithmetic logic. [13] includes further improvemens to the start-gap technique.

The start-gap technique subdivides the memory space into a number of lines (generally $N = 2^n$ total lines: $2^n - 1$ active lines and one spare, or gap, line). The gap line begins at the last logical address in the array, while the start line initially points to the head line in the memory array. Every G, or "gap write interval", number of writes, the scheme swaps the gap line with the line logically before it in the array. Every subsequent G writes, the gap line continues this pattern.

Once the gap line reaches the top of the array, the start line points to the next logical line in the array, and the gap line swaps with the bottom line in the array. With this in mind, the logical to physical address (PA) translation for an access to logical address LA becomes simple arithmetic: If $(LA + start)\%N >= gap$, then $PA = (LA + start)\%N + 1$; otherwise, $PA = (LA + start)\%N$.

Chapter 3

The PCM DIMM

In place of the DDR2 DIMMs on which Moneta stores data, Onyx uses a set of custom-designed PCM DIMMs. This chapter describes the PCM used, the PCM DIMM architecture, problems with the current revision of the PCM DIMM, and potential changes for future PCM DIMMs.

3.1 The PCM Module

The PCM DIMM design uses Micron's first generation *P8P* 16 MB PCM devices (part number NP8P128A13B1760E) [2]. The PCM provides a NOR-flash-like interface to the array. The devices provide 23 address bits, 16 data lines, and a number of other interface lines (e.g. write enable, output enable, and chip enable). To issues writes on a device with a NOR-like interface, a controller should send a command and an address over the device's IO lines, followed by a series of data chunks. For a read, a controller should send a command and an address and wait for the device to complete the read operation. Once the read has completed, the controller should read the data in chunks from the device. Also, in NOR flash devices, a device must perform an erase on an entire flash block before individual cells may be programmed. However, unlike NOR flash, the PCM devices allow for arbitrary bit-level writes to the memory array, which does away with flash-style erase commands. The P8P device exposes various commands (e.g. read array, write, buffered write) through which reads and writes take place. Future PCM

devices should not be restricted to the limited NOR-flash-like interface that the P8P parts provide, as it is not a limitation of the PCM technology itself.

3.1.1 P8P Command Interface

Reads to the P8P PCM are asynchronous. In order to read data from the device, a controller must first place the device into *read array* mode, which primes the PCM device to accept read commands. The controller should then set the address lines, which places 16 bytes of data into an internal buffer. The controller then toggles the output enable signal to stream the data from the internal buffer.

Writes are available in two forms: 16-bit *writes* and buffer-sized writes (or *buffered writes*). Onyx uses buffered writes as its primary write mechanism because buffered writes provide higher bandwidth to the PCM array. Compared to *writes*, buffered writes require a more involved sequence of events. First, a controller must send the buffered write command to the device and poll the status register of the device until it has completely initialized its internal buffer. Then, the controller sends a metadata packet followed by up to 64 bytes (one buffer's worth) of data. After the device receives a "buffered write start" command, it begins writing the buffer to the PCM array. Again, the controller must poll the device's status register to determine when the write has completed and the data has been completely committed to the non-volatile array.

3.1.2 P8P Performance and Durability Characteristics

Theoretical maximum bandwidths and minimum latencies for both the read and buffered write commands are reported in the P8P datasheet: Reads require 314 ns for every 16 bytes and buffered writes require 120 μs for every 64-byte buffer of (random) data. PCM write latency varies based upon the actual data written. The above latencies estimate the maximum read and buffered write bandwidths per device of 48.6 MB/s and 0.5 MB/s, respectively.

Like other recent non-volatile memories (such as flash), PCM devices have a tendency to wear out, and cause data corruption on both writes and reads. The

P8P datasheet states that these devices have an average lifetime of one million write cycles before the first bit error will appear.

3.1.3 P8P Voltage and Power Requirements

Each P8P PCM device operates between 2.7 and 3.6 V. IO voltages to the device must be within the range of 1.7 and 3.6 V. According to the datasheet, a P8P PCM device can draw up to 50 mA. This only occurs during write operations, as read operations only require 42 mA and a device in standby requires only 160 μA. While high current usage for operations appears to be normal for PCM devices, the higher voltage requirement does not appear to be.

3.2 BEE3 Limits

Pin assignments, board dimensions, and power requirements are the most important design constraints that I budgeted for in the DDR2 DIMM replacement design. This section outlines the specifics of each constraint.

Pin Assignments

Each DDR2 socket in the BEE3 exposes 240 pins to the attached board. Of the 240 pins: 64 ground pins, 11 power pins, 11 IO power pins, one reference voltage pin, six I^2C pins, one reset pin, and 11 reserved pins make up the set of usable pins in the replacement card design. 135 pins remin for use as IO. The two DDR2 DIMMs share 130 of those pins. The remaining five pins uniquely connect to each DIMM in the pair.

Board Dimensions

The BEE3 enclosure only allows for a card height of just over 46.5 mm. Also, the DDR2 socket imposes a number of dimension constraints on the design of a DDR2 DIMM replacement card: Each card must measure exactly 133.35 mm wide and 1.27 mm thick to meet the criteria of the DDR2 socket.

Power Requirements

A conventional ATX power supply unit–the Thermaltake TR2-4800W–provides power for the BEE3 system [4]. In order to accurately estimate the available power budget available to each DIMM, I traced through the multitude of power systems that exist throughout the BEE3. The BEE3 manual provides more information about the BEE3 power distribution network.

Power to the DIMM sockets originates from one of two 12 V/168W rails on the PSU. Each 12 V rail provides power through three TI PTH08T210W power modules. One of these modules provides 12 V to 1 V conversion to the two FPGAs powered by this 12 V rail. The remaining two modules provide 12 V to 1.8 V conversion to both the FPGAs and the eight DIMM sockets–each module powering one FPGA and four DIMM sockets.

From estimates in the BEE3 manual, the first power module uses 28.56 W to power the two FPGAs. This leaves 139.44 W for the two remaining power modules. Further estimates from the BEE3 manual point to a power conversion efficiency of 88.9% for each of the 12 V to 1.8 V power modules. This yields a total of 41.994 W of available power for one FPGA with its DIMM sockets. The FPGA on this rail could require up to a maximum of 10 W power from this power subsystem. This leaves the four DIMM sockets with a power budget of 31.994 W.

3.3 Logical PCM DIMM Description

The two major goals of the PCM DIMM design were to maximize both the read/write bandwidth and the capacity per board. This section discusses the former while the next secion will elaborate on the latter. Since the P8P PCM chips perform better than flash memory for reads and on the order of flash memory for writes (120 μs per operation from Section 3.1.2), the PCM DIMM design optimizes for write performance. To accomplish this, the design must maximize the number of devices actively written to at any given time.

One design that satisfies this goal divides the available chips into groups which execute an operation simultaneously, where all of the devices in each group

share chip enable lines. I refer to devices grouped in this fashion as a *rank*. The P8P PCM module device architecture provides a chip enable for each device, where a device will only accept commands when a controller asserts this signal.

To simplify the PCM DIMM interface, I limited the data interface width– or rank width–to a power-of-two devices. This reduces the complexity of striping through Onyx, and easily maintains a contiguous address space throughout the system. Also, since data integrity issues and wearout plague PCM, the PCM DIMM design must include storage for error correcting codes, or ECC, and other metadata. With the above restrictions in mind, a PCM DIMM rank must contain a power-of-two number of devices plus an additional device for ECC and other metadata, or $2^M + 1$ devices per rank (where M is defined below). Analysis in Section 3.4 shows that a designer may place a maximum of eight ranks per PCM DIMM.

In order to determine the number of chips that can operate in each rank, the design must consider the number of total signals that must interface with the host FPGA. All the chips on the pair PCM DIMMs share the following signals from the FPGA: output enable, write enable, write protect, reset, and power good. Since the design calls for independantly controllable ranks, each of the eight ranks require a chip enable signal. Each DIMM uses an additional pin as a DIMM enable. Also, each P8P PCM module also exports 23 address bits. Of the 135 pins, 99 pins remain for the data interface. Since each P8P PCM module has 16 pins, M has an optimal value of two (in $2^M + 1$ above). This requires an additional 80 pins. Eight LEDs and a collection of forwards-compatible pins (for the next-generation Micron PCM) use the remaining 19 pins. I discuss the next-generation Micron PCM pins in Section 3.7.

3.4 Physical PCM DIMM Description

3.4.1 Physical Dimensions

With the physical design constraints from Section 3.2 in mind, each PCM DIMM is 46.5 mm tall, 133.35 mm wide, and 1.27 mm thick (not including part

thicknesses). These dimensions–minus reserved space for the DDR2 DIMM socket pins, power converter circuitry, and other periphery– yielded enough space for 40 total chips per side. I did not populate both sides of the PCM DIMM because of the additional routing complexity and strain on the PCM DIMM power system. Likewise, the cost of fabrication for the PCM DIMM would have become prohibitively expensive (because of the added routing layers to accomadate the additional chips). Figure 3.1 shows a picture of the PCM DIMM.

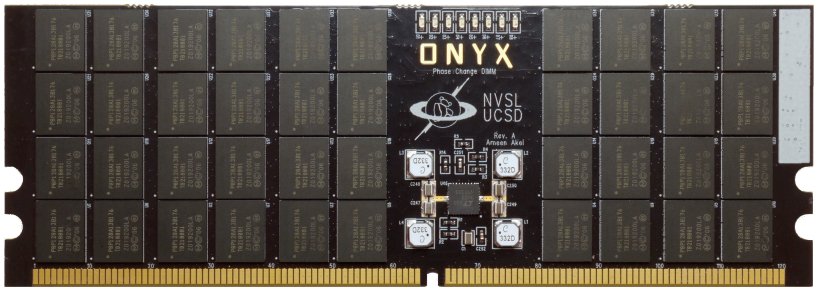


Figure 3.1: This is an image of a PCM DIMM. Each DIMM contains 40 P8P PCM parts, a boost converter, among other peripherals. Each DIMM contains 640 MB of non-volatile storage.

3.4.2 Peripheral Parts

The PCM DIMM also integrates a number of different parts to either support the function of the P8P PCM or perform other auxiliary tasks. The following sections describe the peripherals in more detail.

Capacitors

Each PCM DIMM contains a number of different capacitors: small decoupling capacitors, medium decoupling capacitors, and very large atomic-write capacitors. The small decoupling capacitors (0.1 μF) are evenly distributed across the power plane on the back side of the PCM DIMM, at a frequency of six capacitors per PCM module as recommended in the P8P PCM datasheet. The medium decoupling capacitors (4.7 μF) are likewise distributed at a rate of one for every six

PCM devices. The atomic-write capacitors provide enough power for any number of writes on a particular PCM DIMM to complete in the event of power loss. The amount of capacitance provided by these capacitors totals 2.82 mF. The model for current usage per PCM device used to derive this value models a simple RC circuit with the worst-case current usage and buffered write operation time from the P8P datasheet. As of this writing, the effectiveness of the atomic-write capacitors have yet to be verified in a real system experiment.

Boost Converter

In order to provide conversion from the 1.8 V provided by the DDR2 DIMM socket to the 3 V required by the P8P PCM, I use the LTC3425EUH four-phase boost converter from Linear Technologies [10]. The converter requires four external inductors and a number of resistors and capacitors to perform its function. Section 3.5 provides more detail on the power conversion systems in the PCM DIMM.

Other ICs and LEDs

The design employs a I^2C EEPROM to store identification information on a per-PCM DIMM basis. The EEPROM connects to the reserved I^2C data lines in the DDR2 DIMM standard. The PCM DIMM also employs two OR gates to control its chip enable lines. Each set of OR gates interfaces with a DIMM enable and a chip enable, so that a chip enable assertion will only apply to the PCM DIMM that is currently selected (since chip enable lines are shared amongst PCM DIMMs in a pair). A set of eight green LEDs are connected directly to the FPGA via PCM DIMM pins. Some of these LEDs are connected to shared pins while others are connected via DIMM-unique pins.

3.5 Power System

Since the P8P PCM chips require a significant amount of power, power conversion for the PCM DIMMs is a non-trivial task. As noted in Section 3.2,

each DDR2 DIMM socket provides 1.8 V. Also, Section 3.1.3 shows that each P8P PCM device requires a voltage between 2.7 and 3.6 V. The design uses a voltage of 3.0 V on the recommendation of Micron engineers, so as to strike a balance between high performance and higher reliability. Section 3.1.3 also defines the current usage of the P8P PCM module as no greater than 50 mA. I use this estimated current throughout the power system calculations in order to provide conservative estimates for the power budget.

Given the operating voltage and estimated current usage of a P8P PCM module, each module requires no more than 165 mW of power. Across 40 chips, each PCM DIMM requires up to 6.0 W of power (or 2 A at 3 V). According to the LTC3425EUH datasheet, the IC will operate at an approximate efficiency of 88.5% when converting from 1.8 V to 3.0 V at 2 A. This requires 6.78 W at the input of the boost converter in order to satisfy its power requirement. My analysis ignores the other subsystems in the PCM DIMM because of their relative insignificance when compared to the power usage of 40 P8P PCM modules (< 10 mA at 3.0V). So, for a four-DIMM group, the total power required is 27.12 W. This meets the power budget of 31.994 W available at the combined inputs of the DDR2 DIMM sockets.

The design of the PCM DIMM power system was one of the more significant challenges in developing Onyx. Because the PCM devices draw a large amount of current (and therefore a large amount of power), the power system needed to provide enough current at the right time for all of the operating devices without becoming unstable or lowering the operating voltage below the operating range of the PCM devices. I overcame the challenge through exhaustive modeling of the proposed power system through Linear Technology's LTSpice circuit simulator [11] and through real-world testing with a mock PCM DIMM modeled with passive resistors.

3.6 Potential PCM DIMM Improvements

Throughout the testing phases of the PCM DIMM bringup, I suggested a couple of design improvements. Most are not critical features, but aid in initial board bringup. One suggested improvement provides a method for current measurement on the PCM DIMMs themselves. Although this addition is not trivial, it is a feature that may prove to be useful in future experiments. The second improvement exposes regular test points for all PCM DIMM signals. This helps with both bringup, and helps debug non-functional boards.

3.7 Changes for New Micron PCM

New 45 nm PCM from Micron Technologies will bring a number of welcome changes that affect the PCM DIMM design: lower supply voltages (and thereby lower power), higher capacity, and synchronous reads. Characteristics of the new 45 nm PCM come from a preliminary datasheet provided by Micron [1].

3.7.1 Lower Supply Voltages

The 45 nm generation of PCM will operate within the supply voltage range of 1.7 to 2.0 V, in contrast with the 2.7 to 3.6 V range of the P8P PCM modules. Since the BEE3 supplies 1.8 V at the DDR2 DIMM socket, the 45 nm PCM will not require voltage conversion. This eliminates the boost converter (or any similar IC) from the next-generation PCM DIMM design. The complexity savings will afford cheaper board manufacturing costs, and the space savings will allow for potentially more PCM on the board.

3.7.2 Higher Capacity

A P8P PCM module only provides 128 Mb worth of storage; however, a 45 nm PCM module providess 1 Gb worth of storage. This will increase the amount of storage per PCM DIMM by 8×, without any changes in module count. This will increase the total capacity from 640 MB to 5 GB per PCM DIMM. This also

impacts the physical design of the PCM DIMM: 45 nm PCM requires 26 address bits versus the 23 in the P8P PCM modules. This addition already exists in the current version of the PCM DIMM.

3.7.3 Synchronous Reads

45 nm PCM also brings about faster, synchronous reads. This requires additional signals: address valid, clock, and wait. These signals already exist in the current PCM DIMM design; however, the next designer should examine the merits of length-matching these traces in the next PCM DIMM revision.

Chapter 4

The Onyx PCM SSD

The Onyx system operates almost identically to the Moneta system described in Section 2.1. The increased number of transfer buffers (from eight to 16) and the memory controllers (DDR2 DRAM to PCM) make up the only major differences between the Moneta SSD and Onyx SSD designs. Figure 4.1 shows an overview of the modified Moneta design that comprises Onyx. I use the remainder of this section to describe the Onyx memory controller.

4.1 The Onyx Memory Controller

Onyx employs a custom PCM memory controller which translates requests received from the ring and schedules them on the correct PCM DIMM rank. Here, I describe the inner workings of the Onyx PCM memory controller.

4.1.1 Functional Description

Of the four PCM DIMMs connected to each FPGA, a single memory controller controls two. Figure 4.2 shows the internal architecture of each of the two memory controllers in each Onyx FPGA.

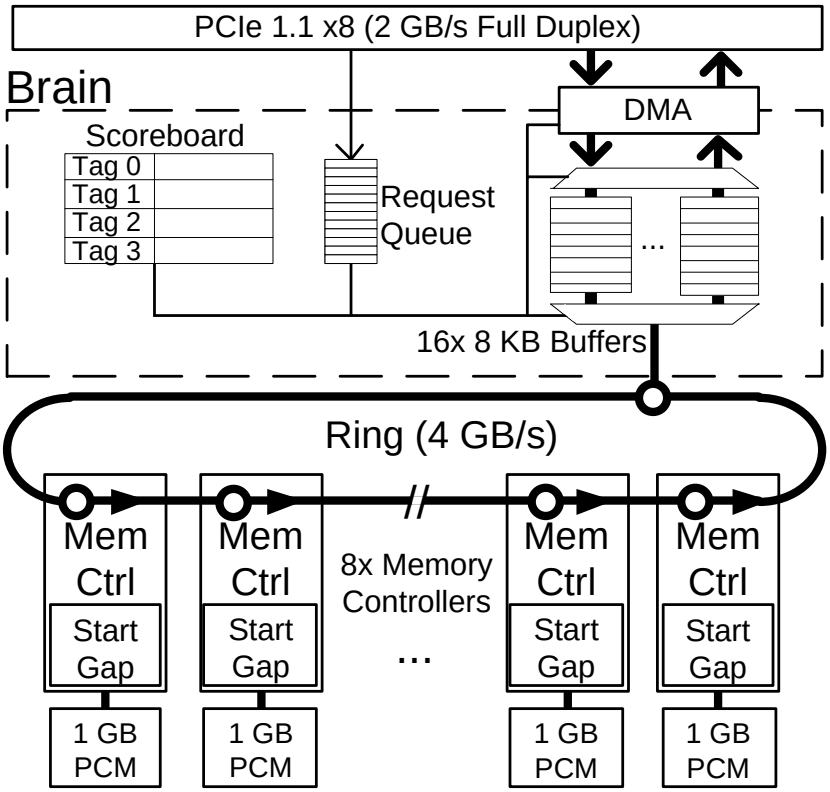


Figure 4.1: This shows the modified version of Moneta's high-level architecture which comprises Onyx. It is comprised of the *brain* module, a ring network, 16 transfer buffers, and eight memory controllers.

Start-Gap Module

The ring interface receives requests over the ring. That module passes the requests to the start-gap wear-leveling module, which applies the principles described in Section 2.2. As a result of the discussion in Section 5.2, I set the start-gap line size in Onyx to 4 KB, and G, or the "gap write interval", to 128 writes.

The start-gap module also breaks up the requests receieved over the ring into 64 byte chunks and passes the address and data components of each chunk on to the next stage through the address and data FIFOs shown in Figure 4.2.

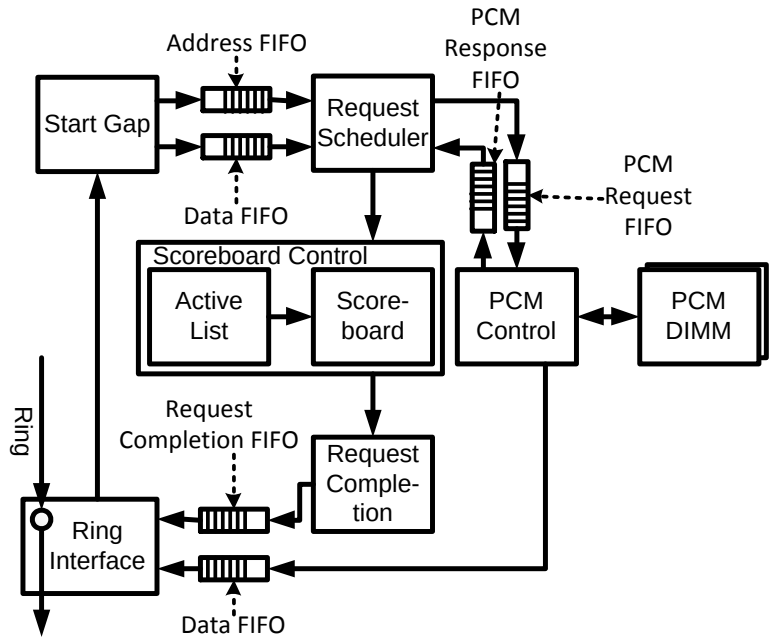


Figure 4.2: This is a high-level depiction of Onyx's PCM memory controller. This design allows Onyx to maximize write parallelism for an access pattern which targets all ranks.

Request Scheduler

The request scheduler is the heart to the Onyx memory controller design. It translates addresses to the proper locations in the PCM DIMM arrays, schedules requests among the available ranks of PCM, and also tracks, queries, and completes outstanding write requests.

A read request in the memory controller does not require much state: The request scheduler sends out a read request to the appropriate rank(s) and processes the next request. However, a write request is more complex. Upon receiving a write request, the request scheduler marks the requisite rank(s) associated with the incoming address as busy in the scoreboard. The active list structure then stores a pointer to the current request. The request scheduler streams the data associated with this request from the incoming data FIFO and the addresses from the address FIFO to the PCM control module to a specific rank in the PCM array.

Since the request scheduler keeps track of requests at rank granularity, it

processes requests on up to 16 ranks at any given time (with the current PCM DIMM design). This allows the request scheduler to overlap the long PCM write latency with other requests. This optimization allows Onyx to achieve its write bandwith performance.

The request scheduler also tracks and generates status requests, which poll the PCM ranks (through the PCM control module) for the status of their write requests. A small amount of state tracks which ranks have outstanding status requests. This logic sends out status requests for ranks that both have a write outstanding but no outstanding status check. This ensures that each rank is consistently checked for completion. When a status check makes its way back through the PCM response FIFO, the request scheduler checks to see if all of the chips on a given rank have finished their write requests. If there are no outstanding writes in a given rank, the request scheduler clears the corresponding state in the scoreboard.

Scoreboard Control Detail

Though the scoreboard appears above as a single structure, the underlying pieces are more sophisticated. Outstanding write request metadata storage requires two-cycles, and involves three data storage structures: the scoreboard, the write table, and active/free lists. Figure 4.3 demonstrates this relationship.

The write table stores data on a write-request basis: Each entry in the write table represents the number of outstanding writes to individual PCM ranks. A zero in an active entry represents a completed write request. The scoreboard contains an entry for each rank abstracted by the PCM DIMMs (16 entries in this revision of Onyx). Each write request represented in a write table entry is comprised of writes to one or more ranks, or one or more entries in the scoreboard. With that in mind, each entry in the scoreboard contains a pointer to its "parent" write table entry.

Here is a short illustration of how the scoreboard functions. When the request scheduler processes a write, a pointer to the first free entry in the write table (read from the free list and transferred to the active list) is stored into the

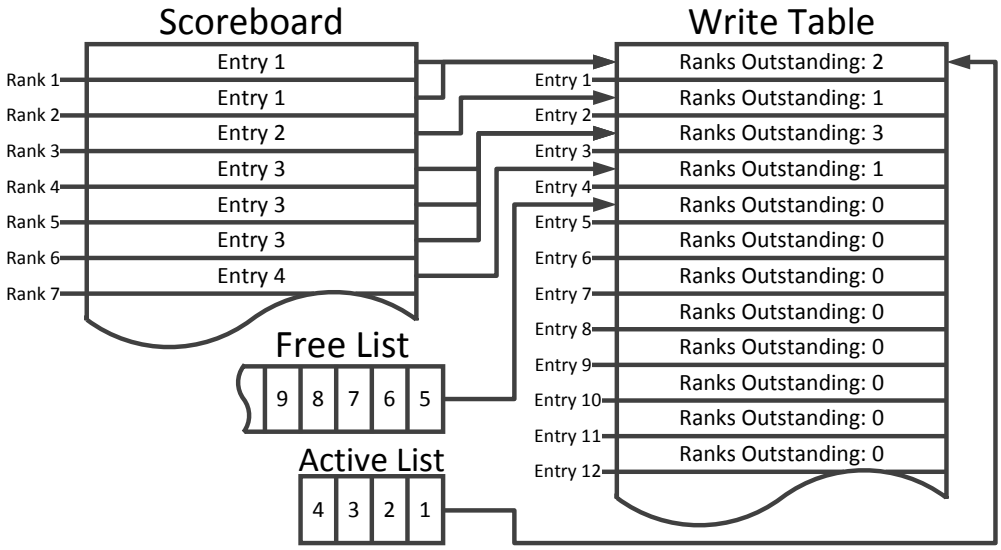


Figure 4.3: This details the Onyx memory controller's scoreboard. The scoreboard tracks requests on each rank, while the write table tracks outstanding write requests. The active list stores active entries in the write table, and the free list stores empty entries.

scoreboard entry for the rank that will process the first part of this write. On the following cycle, the scoreboard state machine increments the allocated entry in the right table to signify that one rank is processing the data for this request. If this write request spans more than one rank, the state machine repeats that process for each rank involved in the request.

When a request completes, the request scheduler clears an entry in the scoreboard, which signals the scoreboard control logic to decrement the proper write table entry (pointed to by the reference in the soon-to-be-cleared scoreboard entry). This process continues until the write table entry for a particular request reaches zero, at which time the request completion module completes the request.

Request Completion

Onyx allows for two different methods for write request completion: early and late. Early completion allows for a write to be signaled as complete (back to

the Onyx brain, and thereby the host system) as soon as the entire request has been read out of the FIFOs in the request scheduler. This allows Onyx to overlap PCM write times with the request latency through the system. The Onyx system guarantees that all requests for which it generates write completions will be written to the PCM array in the event of any power loss through the use of large system-wide and PCM-DIMM-wide capacitance. The late completion scheme completes requests only when a write request has been written to the PCM array.

The request completion module both frees completed requests from the write table and generates write completion signals for the late completion scheme. This module montiors the write table (in the order that requests were sent to the memory controller). When all of the ranks processing the parts of a write request have completed (and the relevant write table entry reaches zero outstanding ranks), the request completion module transfers that entry in the write table from the active list to the free list. It is at this point that the Onyx memory controller generates a late write completion request.

PCM Control

While the request completion module generates requests intended for the PCM array, it only generates high-level requests and does not deal with the intricacies of communicating with the PCM. The PCM control module handles this communication.

Requests destined for the PCM array come through the PCM request FIFO. The PCM control module translates these requests into low-level requests sent directly to the PCM. This module also verifies that the PCM array is ready for data on a buffered write operation and communicates status request results back through the PCM response FIFO. Lastly, the PCM control module routes data read from the PCM array for read requests back to the ring interface.

4.1.2 Design Improvements

A few design improvements may potentially increase the performance or resource usage of the Onyx memory controller and the Onyx design as a whole:

increased clock speed from 125 MHz to 250+ MHz, out-of-order completions, early completions at the Onyx brain, background erasing for faster write performance, and write suspends.

Clock Speed Increase

The current memory controller design for Onyx runs at 125 MHz. An increase in clock speed of the Onyx memory controller increases the total read bandwidth overall as a result. 8 ns (125 MHz) is the current cycle time of the design. A single page-mode read in the current generation of PCM requires 25 ns per chunk of data; however, the current memory controller design can only read at 32 ns increments (without data corruption). This wastes 7 ns per chunk of data read. An increase in clock speed to 250 MHz decreases the wasted cycle time to 3 ns–which results in a significant increase (around 100 MB/s for large reads) in read bandwith for large accesses. Small read operations will see a less-significant improvement in bandwidth. This increase in clock speed also benefits writes and status requests, however, marginally. This improvement requires partitioning the request scheduler state machine, as it is the most complex part of the memory controller design.

Out-of-order Completion

Out-of-order completion support is a more daunting task, and would not provide a significant benefit in the presence of the early completion request strategy. It requires a number of interface changes between the start-gap module and the request scheduler, which allows for write completion notifications to be sent out of order (the current design only allows for write completion requests to be generated in the order that requests are sent to the request scheduler). Once this interface can accomodate out-of-order completion notifications, the request completion module must be modified to take into account all active entries in the write table instead of solely the oldest active entry.

Brain Completion

Brain completion support allows the system to mitigate more overhead (ring transfer time, etc.) by allowing new requests to be sent through the system sooner. The design contains a version of this change, however, it does not yet function correctly. Changes required for this improvement would be more implementation rather than architecture-based.

Background Erases

While PCM, unlike flash, does not require erase operations between writes or programs, erased areas of the PCM array program faster than non-erased areas. Writes on erased areas of the 90 nm PCM devices require (typically) 41% less programming time than their non-erased counterparts. This gap deepens for 45 nm PCM, as writes on erased areas require 58% less programming time. There are a few disadvantages to this improvement: If applied too aggressively, this technique exposes a significant amount of extra wear on the PCM (in the worst case, half of the lifetime will be consumed by erases). Also, if this technique is executed just before an incoming write, it can potentially increase the write latency of a request (though use of the erase suspend command may mitigate the negative impact of this scenario).

Background erases require changes to the request scheduler, as that module would best know when to optimally execute an erase. Also, potentially, the design can employ prediction heuristics to determine the best time in which to execute a background erase.

Write Suspends

Because of the long write latencies for current-generation PCM, mixed (read and write) workloads can suffer if a read follows a write to the same PCM DIMM rank (but not to the same location). The Onyx system can suspend ongoing writes in order to service other requests, like reads in order to lower read latencies. This paper mentions a similar technique [12]. Write suspends allow Onyx to lower the read latency of requests in specific cases. The new 45 nm PCM limits the benefits

of this improvement with the introduction of partitions and simultaneous interpartition operations (Section 4.1.3). In order to fully implement write suspends, the request scheduler needs the ability to determine when a write suspend should be initiated, and the PCM control module must either stall a request until the suspend goes into effect or communicate the status of the suspend back to the request scheduler.

4.1.3 Changes for New Micron PCM

The new Micron 45 nm PCM includes a number of new features, many of which Section 3.7 addressed at the PCM-DIMM-level. This section addresses the PCM changes that will affect the Onyx memory controller design: larger internal write buffers, simultaneous interpartition read/write operations, and synchronous read operations.

512-byte Internal Write Buffers

The new Micron PCM includes larger 512-byte write buffers (compared with 64 bytes). The buffers allow for more data storage per buffered write operation. This highlights a shortcoming in the Onyx memory controller: Without changes, the smallest possible write request becomes 2 KB (instead of 256 bytes). This problem exists because of the interface currently exposed to the Onyx memory controller. Requests enter the memory controller as 32 byte increments, and no indication is given of the size of the request as a whole. So, in order to support buffered writes, the memory controller must assume that the current request will satisfy a minimum of one rank's worth in buffered writes. This is not a problem with the 90 nm PCM, as the minimum write size is 64 bytes-per-buffer across four chips per rank (or 256 bytes); however, with the new 90 nm PCM, the minimum write size becomes 512 bytes-per-buffer across four chips per rank (or 2 KB). The problem is exacerbated by the fact that the PCM write protocol requires the size of the buffered write to be communicated to the PCM before any data is sent to the PCM. A couple feasible solutions exist for this problem: restructuring the interface into the request scheduler so that it communicates the size of the current

request or buffering one request's worth in data in the request scheduler before it issues a write command to the PCM. Resource and performance-conscious design points to the former solution.

Simultaneous Interpartition Read and Write Operations

The new 45 nm PCM partitions its storage into 16×64 Mb blocks. The device also allows for two simultaneous operations to occur on different partitions, with some restrictions. The most useful set of operations allows for both a read and a write to occur simultaneously. The request scheduler and scoreboard logic will need to be aware of differing partitions and need to be able to schedule both reads and writes on the same rank (given the correct circumstances).

Synchronous Read Operations

The next generation of PCM exposes a faster, synchronous interface from which to read data from the PCM array. This will allow for faster reads, as long as the reads satisfy the burst-length (at least 4, 8, or 16 words) and alignment constraints. The largest burst-length (16 words across four devices per rank): 128 bytes, does not pose the same problem that exists in the larger-write-buffer case above. The main changes required to properly support synchronous writes are limited to the protocol-laden PCM control module, as it needs to support the new ADV, WAIT, and CLK signals.

Portions of the content in Chapter 4 appeared in the Proceedings of the 3rd USENIX Conference on Hot Topics in Storage and File Systems, 2011. A. Akel, A. M. Caulfield, T. I. Mollov, R. K. Gupta, and S. Swanson. Onyx: A Prototype Phase Change Memory Storage Array. The thesis author was the primary investigator and author of this paper.

Chapter 5

Onyx's Performance

This section uses both a microbenchmark and a real-world application to measure Onyx's performance. I also use the microbenchmark to tune striping performance of Onyx, to understand the CPU usage of Onyx, and to characterize the performance impact of start-gap wear-leveling in Onyx.

5.1 Raw Performance

Figure 5.1 shows the results of the microbenchmark XDD [16] on Onyx compared with the results from FusionIO's 80 GB ioDrive [8] flash SSD and the Moneta SSD. XDD performs reads and writes to a raw device, and measures the bandwidth of that device as a function of request size. In Figure 5.1, the results show the bandwidth of a particular device as a function of both request size and benchmark type (100% reads, 100% writes, and 50% reads/writes). Table 5.1, Table 5.2, and Table 5.3 show the results from Figure 5.1 for reads, writes, and read/writes (respectively) in a tabular format. The XDD benchmarks all employ 16 IO threads, unless otherwise specified.

For reference, the measured random read and write latencies for the ioDrive are 36 μs and 60 μs, respectively. For the Moneta device the measured random read and write latencies are 13 μs and 14 μs, respectively.

The results show that Onyx sustains over 1.1 GB/s for large-request-size/device-bandwidth-limited reads and over 340 MB/s for writes with early completion no-

Graph 5.1: This graph depicts the results of a bandwidth versus request size study, which compares Onyx (with both early and late completion methods), Moneta, and the FusionIO ioDrive.

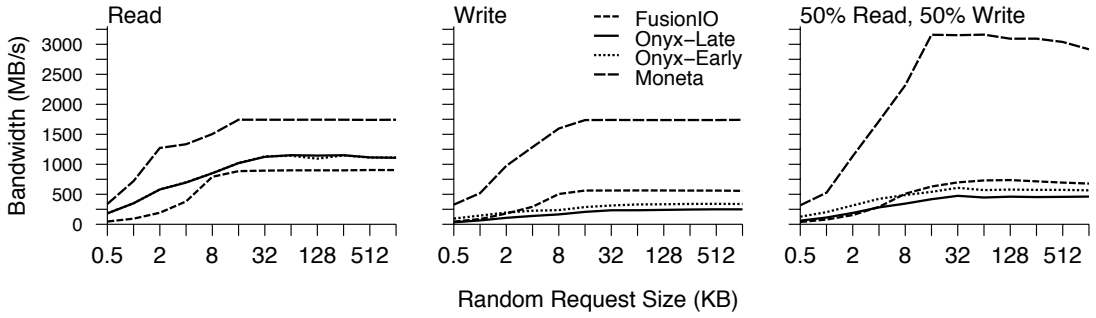


Table 5.1: This table demonstrates the read bandwidth results shown in Graph 5.1. Onyx outperforms the ioDrive in terms of read bandwidth across all request sizes (by between 7% and 300%). Moneta outperfoms both.

Access Size (bytes)	Read Bandwidth (MB/s)		
	Onyx	FusionIO	Moneta
512	183.865	46.074	338.157
1024	350.364	93.422	720.615
2048	581.633	192.413	1272.399
4096	701.020	381.316	1334.618
8192	851.713	791.569	1504.561
16384	1022.392	885.925	1743.099
32768	1131.279	894.507	1742.896
65536	1142.802	899.541	1742.348
131072	1094.287	899.364	1743.424
262144	1147.993	899.087	1742.788
524288	1112.242	904.800	1740.112
1048576	1114.176	903.606	1742.847

Table 5.2: This table demonstrates the write bandwidth results shown in Graph 5.1. The Onyx-Early results show a significant improvement over Onyx-Late. Onyx-Early outperforms the ioDrive for writes up to 2 KB, the reverse occurs for larger requests.

Access Size (bytes)	Write Bandwidth (MB/s)			
	Onyx-Late	Onyx-Early	FusionIO	Moneta
512	33.398	95.384	44.531	327.433
1024	64.393	145.077	87.213	518.534
2048	107.629	196.384	179.637	974.390
4096	139.072	226.045	291.704	1286.862
8192	164.643	236.118	504.485	1595.978
16384	206.857	287.123	562.643	1737.253
32768	233.694	313.177	563.129	1739.333
65536	234.023	330.814	564.271	1737.388
131072	240.230	333.963	563.587	1737.372
262144	246.084	339.665	561.792	1737.772
524288	249.590	340.820	561.918	1737.508
1048576	248.879	338.282	557.153	1741.268

Table 5.3: This table demonstrates the bandwidth results for a 50% read/write workload shown in Graph 5.1. The results of the read/write workload show similar trends to those in the write-only workload.

Access Size (bytes)	50% Read/Write Bandwidth (MB/s)			
	Onyx-Late	Onyx-Early	FusionIO	Moneta
512	60.775	126.421	38.871	316.787
1024	112.733	202.849	78.639	526.501
2048	189.761	314.063	153.875	1135.785
4096	281.868	424.427	290.316	1720.071
8192	344.614	487.003	509.003	2317.422
16384	417.928	540.693	629.320	3159.317
32768	474.233	607.689	698.708	3152.228
65536	446.594	570.224	731.198	3161.368
131072	460.417	579.057	737.300	3093.404
262144	453.781	573.664	715.835	3094.214
524288	457.365	573.811	694.755	3039.301
1048576	461.622	565.913	678.558	2916.509

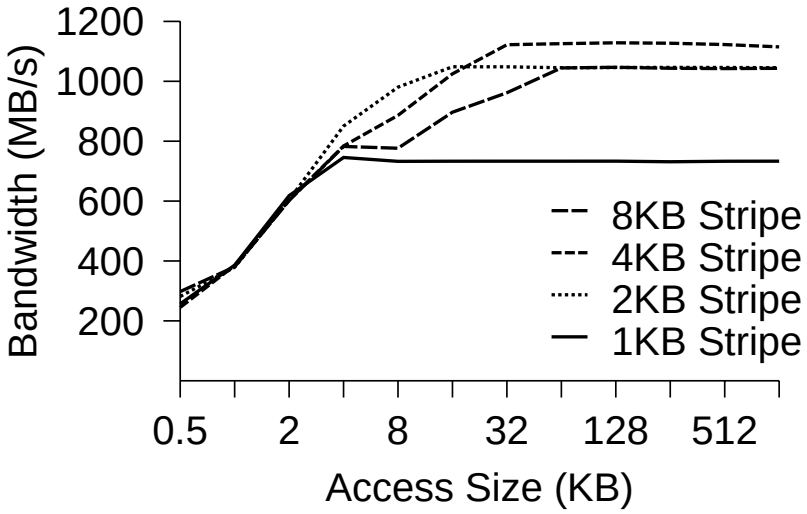
tifications enabled. Given that each PCM device, from Section 3.1.2, theoretically supports up to 48.6 MB/s for reads and 0.5 MB/s for writes, Onyx nearly meets the projected large-request-size read bandwidth of 1.5 GB/s and exceeds the projected write bandwidth of 256 MB/s by up to 34%.

The data also compares the performance difference between early and late write completions. As expected, the completion method does not affect the read performance of Onyx; however, early completion notifications improved write performance by between 32% for large requests and 147% for small requests. The data for the 50% read/write benchmark show similar gains. Because of its superior performance, the Onyx system uses early completion notifications for the remainder of the paper.

The benchmarks also compare the performance of Onyx to that of the FusionIO ioDrive and the Moneta SSD. Onyx outperforms the ioDrive for reads of all request sizes (by between 7% and 300%); however, more diversity exists for write performance. Onyx outperforms the ioDrive for smaller writes (through 2 KB) by between 9% and 114%; however, the ioDrive outperforms Onyx by between 29% and 113% for writes larger than 2 KB. I attribute the performance increase through 2 KB requests to the complex Flash Translation Layer (FTL) that the ioDrive uses, which is not present in Onyx (and which PCM SSDs will not require). I attribute the difference in write performance for larger request sizes to raw write performance differences of the underlying technology in each SSD: The write performance of the late-generation raw flash devices outperforms that of first-generation Micron PCM. With this in mind, I predict that with next-generation PCM from Micron, a future version of Onyx will outperform the ioDrive in both reads (by 2×) and writes (by 5-6×). The data for the 50% read/write performance shows that there is no disadvantage of performing both reads and writes simultaneously in either system, as the results show an average of both pure read and pure write performance. The Moneta results show significant performance improvements versus both Onyx and the ioDrive. Specifically, the Moneta results show an upper bound for future Onyx performance.

5.2 Effects of Stripe Size

Graph 5.2: A 4 KB stripe size provides the best read performance in Onyx. A 4 KB stripe allows for efficient transfer buffer usage and high memory-controller-level parallelism.



This study varies the granularity at which Onyx stripes data across its eight memory controllers. Figure 5.2 shows the results of this study for read bandwidth across request size with XDD. The data show that the optimal stripe size for Onyx with current-generation PCM stands at 4 KB. This result largely depends on the size of the internal buffers in current-generation PCM. For reads, smaller request sizes provide for better read bandwidths; however, because of the transfer buffer space allocation in Onyx, not enough 1 KB requests reach the memory controller to provide the appropriate read parallelism. I expect that smaller stripe sizes will outperform larger stripe sizes as design changes address this shortcoming in the Onyx brain.

Write stripe size selection depends upon the internal characteristics of the PCM: Section 3.1.2 reveals the internal write buffer size of current-generation PCM as 64 bytes. Across 16 ranks of 4 PCM devices per rank, each memory controller commands an internal write buffer size of 4 KB. As mentioned above, the internal write buffer capacity at each memory controller directly influences the optimal stripe size for writes. A stripe size larger than the write buffer size would queue

the remaining data for a request at the memory controller. The additional requests would incur an additional write latency penalty as the PCM devices processed the other write requests. A stripe size less than the buffer size (with the current transfer buffer allocation scheme in Onyx) would waste buffer space. In this case, the system would not queue up enough requests at the memory controller to exploit the maximum amount of parallelism. With this discussion in mind, all of the results presented for Onyx in this paper assume a stripe size of 4 KB.

5.3 CPU Usage

Due to the relative simplicity of Onyx's driver in comparison to that of the ioDrive's, Onyx uses less CPU time per operation than the ioDrive. A system using Onyx uses 20-51% less CPU time per IO operation with the XDD benchmark. This reduction in CPU time also yields a savings in total system power. This advantage also increases the availablity of CPU time for other computation in the system.

5.4 Effects of Start-Gap

Graph 5.3: Start-gap interval selection has a significant impact on bandwidth, because it interferes with request parallelism, while its impact on latency is less significant.

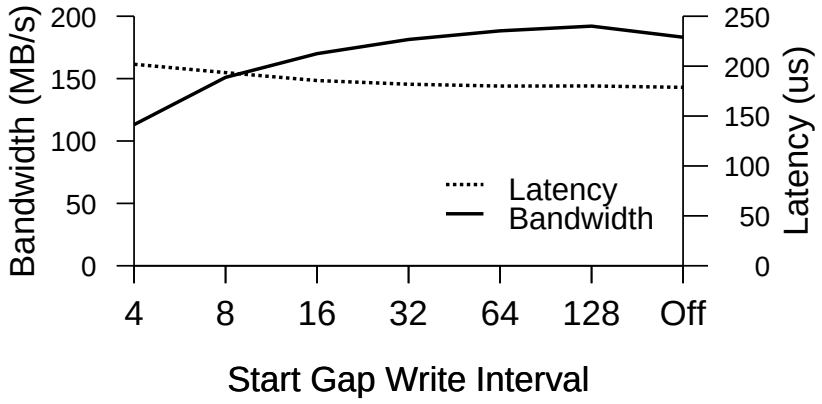
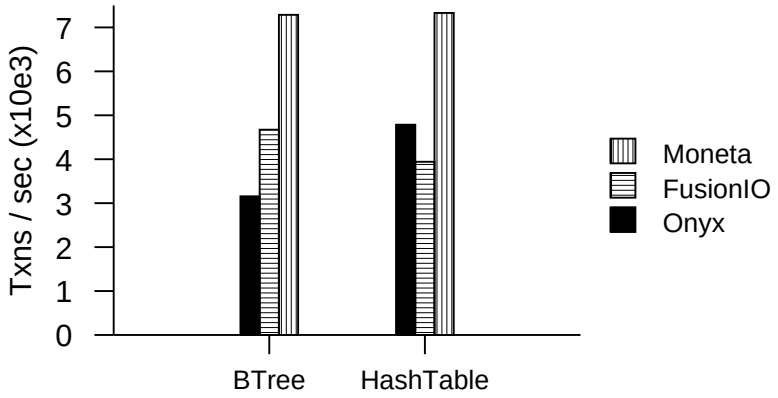


Figure 5.3 shows the effect of varying G (the gap write interval) on both write latency and write bandwidth with XDD. The x-axis varies the gap write

interval from four writes per gap move to 128 writes per gap move. The last data point on the x-axis shows the result of each test while start-gap is off. The write latency data (right axis) shows results using a single XDD thread performing 4 KB accesses to a single memory controller. The write bandwidth data (left axis) shows results for a 16-threaded XDD run across the entire Onyx device. The effect of start-gap on the latency of a single write request in Onyx is minimal: The write latency only varies by 23 μs across the gap write interval sweep. This shows that write latency through the Onyx software and hardware stacks overshadows the latency penalty associated with a gap move. However, start-gap significantly affects Onyx's write bandwidth: The data show that bandwidth can vary by up to 80 MB/s from a gap write interval of four to a system without start-gap. From this data, a system designer can deduce that selecting a gap write interval as small as 32 writes will minimally impact the performance of Onyx (around 10 MB/s in terms of bandwidth and a negligible impact on latency). As defined in Section 4.1.1, the Onyx system assumes a gap write interval of 128.

5.5 Real-world Applications

Graph 5.4: The BerkeleyDB results show real-world performance of each system under test: Onyx, the FusionIO ioDrive, and Moneta.



This section measures the real-world performance of Onyx versus the ioDrive and Moneta with a database benchmark called BerkeleyDB (BDB). Figure 5.4 shows the results of two distinct BDB storage benchmarks: HashTable,

which stores its database in the form of a hash table, and BTree, which stores its database in the form of a b-tree. The employed BDB configuration includes full transaction support and performs synchronous IO. The BDB benchmark performs transactional swaps of the values of two keys in the database. The graph shows the results of the best performing thread count for each device: one for the ioDrive, four for Onyx, and two for Moneta. While the ioDrive outperforms Onyx for the BTree benchmark at 48% greater operations per second, Onyx outperforms the ioDrive for the HashTable benchmark by sustaining 21% greater operations per second. Moneta outperforms both the ioDrive and Onyx in both BDB benchmarks, and, as in the raw performance results, shows an upper bound for Onyx with faster PCM.

Portions of the content in Chapter 5 appeared in the Proceedings of the 3rd USENIX Conference on Hot Topics in Storage and File Systems, 2011. A. Akel, A. M. Caulfield, T. I. Mollov, R. K. Gupta, and S. Swanson. Onyx: A Prototype Phase Change Memory Storage Array. The thesis author was the primary investigator and author of this paper.

Chapter 6

Conclusion

Onyx shows the current state of PCM-based storage arrays and allows designers to understand the idiosyncrasies of PCM as a storage medium. This exploration of first-generation PCM as the storage medium for a fast, non-volatile SSD shows PCM's viability as a flash alternative for certain workloads, and that future-generation PCM will only improve its viability.

While Onyx performs well in many regards, it has yet to overcome many challenges: Future designers must integrate and tune for future-generation PCM, find ways to further increase parallelism to take advantage of PCM's long-latency writes, among other tweaks to the Onyx architecture. However, even with a significant amount of oustanding challenges, Onyx has shown that, with the correct design choices and optimizations, first-generation PCM SSDs are a viable and formidable alternative to state-of-the-art flash memory SSDs.

Bibliography

[1] Micron 45-nm pcm datasheet. http://www.micron.com/.

[2] Micron p8p pcm datasheet. http://www.micron.com/.

[3] A. Akel, A. M. Caulfield, T. I. Mollov, R. K. Gupta, and S. Swanson. Onyx: A prototype phase change memory storage array. In *Proceedings of the 3rd USENIX conference on Hot topics in storage and file systems*, June 2011.

[4] Bee3 hardware platform user manual. http://bee3.beecube.com.

[5] The bee3 platform. http://www.beecube.com/platform.html.

[6] A. Caulfield, J. Coburn, T. Mollov, A. De, A. Akel, J. He, A. Jagatheesan, R. Gupta, A. Snavely, and S. Swanson. Understanding the impact of emerging non-volatile memories on high-performance, io-intensive computing. In *High Performance Computing, Networking, Storage and Analysis (SC), 2010 International Conference for*, pages 1 –11, nov. 2010.

[7] A. M. Caulfield, A. De, J. Coburn, T. Mollov, R. Gupta, and S. Swanson. Moneta: A high-performance storage array architecture for next-generation, non-volatile memories. In *Proceedings of The 43rd Annual IEEE/ACM International Symposium on Microarchitecture*, 2010.

[8] Fusionio iodrive. http://www.fusionio.com/.

[9] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger. Architecting phase change memory as a scalable dram alternative. In *Proceedings of the 36th annual international symposium on Computer architecture*, ISCA '09, pages 2–13, New York, NY, USA, 2009. ACM.

[10] Linear technologies ltc3425 datasheet. http://cds.linear.com/docs/Datasheet/3425f.pdf.

[11] Ltspice circuit iv simulator. http://www.linear.com/designtools/software.

[12] M. Qureshi, M. Franceschini, and L. Lastras-Montano. Improving read performance of phase change memories via write cancellation and write pausing.

In *High Performance Computer Architecture (HPCA), 2010 IEEE 16th International Symposium on*, pages 1 –11, jan. 2010.

[13] M. Qureshi, A. Seznec, L. Lastras, and M. Franceschini. Practical and secure pcm systems by online detection of malicious write streams. In *High Performance Computer Architecture (HPCA), 2011 IEEE 17th International Symposium on*, pages 478 –489, feb. 2011.

[14] M. K. Qureshi, J. Karidis, M. Franceschini, V. Srinivasan, L. Lastras, and B. Abali. Enhancing lifetime and security of pcm-based main memory with start-gap wear leveling. In *MICRO 42: Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 14–23, New York, NY, USA, 2009. ACM.

[15] The ramp project. http://ramp.eecs.berkeley.edu/index.php.

[16] Xdd version 6.5. http://www.ioperformance.com/.