ENABLING INTRA-PLANE PARALLEL BLOCK ERASE IN NAND FLASH TO ALLEVIATE THE IMPACT OF GARBAGE COLLECTION

by

Tyler Garrett

B.S., Shippensburg University, 2016

Submitted to the Graduate Faculty of

the Swanson School of Engineering in partial fulfillment

of the requirements for the degree of

Master of Science

University of Pittsburgh

2018

UNIVERSITY OF PITTSBURGH

Swanson School of Engineering

This thesis was presented

by

Tyler Garrett

It was defended on

November 12th, 2018

and approved by

Jun Yang, Ph.D., Professor
Department of Electrical and Computer Engineering

Jingtong Hu, Ph.D., Assistant Professor
Department of Electrical and Computer Engineering

Samuel Dickerson, PhD., Assistant Professor
Department of Electrical and Computer Engineering

Youtao Zhang, Ph.D., Associate Professor
Department of Computer Science

Thesis Advisor: Jun Yang, Ph.D., Professor
Department of Electrical and Computer Engineering

ENABLING INTRA-PLANE PARALLEL BLOCK ERASE IN NAND FLASH TO ALLEVIATE THE IMPACT OF GARBAGE COLLECTION

Tyler Garrett, M.S.

University of Pittsburgh, 2018

Garbage collection (GC) in NAND flash can significantly decrease I/O performance in SSDs by copying valid data to other locations, thus blocking incoming I/O requests. To help improve performance, NAND flash utilizes various advanced commands to increase internal parallelism. Currently, these commands only parallelize operations across channels, chips, dies, and planes, neglecting the block-level and below due structural bottlenecks along the data path and risk of disturbances that can compromise valid data by inducing errors. However, due to the triple-well structure of the NAND flash plane architecture and erasing procedure, it is possible to erase multiple blocks within a plane, in parallel, without being restricted by structural limitations or diminishing the integrity of the valid data. The number of page movements due to multiple block erases can be restrained so as to bound the overhead per GC. Moreover, more capacity can be reclaimed per GC which delays future GCs and effectively reduces their frequency. Such an Intra-Plane Parallel Block Erase (IPPBE) in turn diminishes the impact of GC on incoming requests, improving their response times. Experimental results show that IPPBE can reduce the time spent performing GC by up to 50.7% and 33.6% on average, read/write response time by up to 47.0%/45.4% and 16.5%/14.8% on average respectively, page movements by up to 52.2% and 26.6% on average, and blocks erased by up to 14.2% and 3.6% on average. An energy analysis

conducted indicates that by reducing the number of page copies and the number of block erases, the energy cost of garbage collection can be reduced up to 44.1% and 19.3% on average.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

PREFACE

I would like to thank Dr. Yang and Dr. Zhang for their mentorship throughout my research explorations, as well as my committee members. I'd also like to thank my family and friends who have always supported me in the pursuit of my dreams. It is my hope that any success I may have will serve as an example to others that it does not matter where you come from, you can achieve your dreams if you pursue them heart, passion, and an unwavering sense of optimism. It is my firm belief that we all have the ability to do the extraordinary.

1.0 INTRODUCTION

NAND flash memory has embedded itself as a cornerstone of a wide range of technologies, supplementing systems that range from Internet of Things sensory and edge devices to smartphones and larger enterprise systems. With the rise of Machine Learning and online services, today's world is more data centric that ever before. Thus, it has become more critical than ever that storage in these systems maximizes its efficiency in performance, endurance, and energy consumption. In particular, this thesis considers NAND flash memory that composes and empowers Solid State Drives (SSDs). With their shrinking cost and higher performance, SSDs have become a staple choice for storage in servers and data centers over the last decade, replacing mechanical Hard Disk Drives [1,10,23]. As the industry and marketplace have shifted towards a heavier reliance on services such as the adoption of cloud computing, it is important to ensure the performance of SSDs keep up with the demand of the system and limit energy expenditure as much as possible.

One of the biggest issues that exists in SSDs is the need to perform garbage collection (GC) in order to reclaim pages that no longer contain useful data. This operation is related to the organization of the internal components and basic operations that occur inside the SSD. During GC, valid data must be moved around in order to prepare for the erase operation. By doing so, these moving pages can create contention between themselves and incoming I/O requests at the various levels of the NAND flash hierarchy, thus degrading performance [3,4,5]. Additionally,

copying valid data to new locations frequently exacerbates the write amplification problem, thereby expending more energy in addition to reducing SSD lifetime.

Currently, SSDs attempt to mitigate the problems associated with GC by taking advantage of parallelism that exists within the NAND flash hierarchy [6,17,24]. The SSD controller is able to dispatch commands to NAND flash chips simultaneously across different channels. To increase parallelism further, advanced commands were introduced to perform the same command on different dies of a chip and different planes of a die [6,11]. Unfortunately, parallelism does not dive any deeper due restrictions of shared die registers and disturbance risks. This limitation is a problem as GC occurs independently based on the plane's capacity status and cannot effectively utilize these commands to increase its efficiency.

In order to combat these issues, this thesis proposes a new advanced command called Intra-Plane Parallel Block Erase (IPPBE) to bring parallelism a step deeper into the NAND flash hierarchy. IPPBE takes advantage of the shared triple-well structure of the plane and erasing procedure to erase multiple blocks within a plane, simultaneously. By making a small modification to the block decoder of each plane and providing additional command addressing solutions, IPPBE is able to be to introduced without significant changes to the SSD. This design has shown to be able to maximize GC efficiency resulting in reduced write amplification, increased SSD lifetime, and saves energy.

The remainder of this thesis is organized in the following manner. Chapter 2 provides an overview of NAND flash memory and the GC contention problem. Chapter 3 highlights prior works and the problems with current GC techniques. Chapter 4 lays out the details of the IPPBE modifications and design. Chapter 5 reviews the experimental setup and results. Chapter 6

demonstrates how IPPBE can be implemented in conjunction with current and proposed designs. Lastly, Chapter 7 concludes this thesis.

2.0 BACKGROUND

2.1 NAND FLASH OVERVIEW

2.1.1 SSD Internal Architecture and NAND Flash Hierarchy

Inside the SSD is an array of NAND flash chips. Several chips are strung together sharing a channel over which read, write, and erase commands, along with their corresponding data, can be sent. The array typically consists of multiple channels that can operate in parallel with each other. Inside each chip are one or more dies. Within each die are two planes (sometimes four) which are divided into blocks, while the blocks are further divided into pages [6]. The SSD is overprovisioned with spare blocks that may be utilized as blocks wear out and can no longer be used. This overprovisioned space is not counted toward the SSDs capacity and therefore not available to the user [22]. Pages reserve a portion of space that is dedicated for metadata and required for Error Correction Code (ECC) [14]. Depending on the size of the pages, sub-pages are also used by buffering writes in an onboard DRAM buffer prior to committing them to storage. The controller is responsible for many different management features inside the SSD including garbage collection, bad block management, and wear leveling. Implemented via firmware, the controller utilizes a Flash Translation Layer (FTL) to keep track of the logical to physical mapping of data [29]. Additionally, the controller runs ECC checks to correct data that

may have been disturbed via writes to adjacent pages, pages read within the block, or simply due to retention loss caused by leaky floating gates [20, 21]. By virtue of the architectural layout, the controller is able to efficiently dispatch commands to various locations in the NAND array in parallel. Figure 1 depicts the NAND array and SSD architecture.

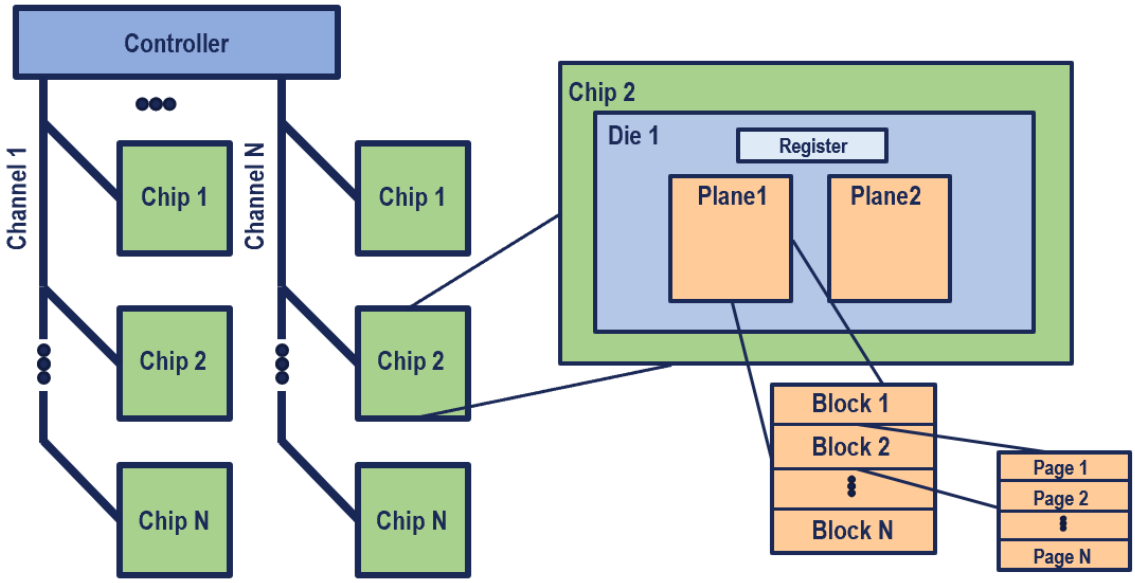

Figure 1: SSD Internal Architecture and NAND Flash Hierarchy

2.1.2 Basic Operations

As previously stated, the controller is responsible for dispatching commands and relevant data to their physical location inside the NAND array. During a write operation, the data to be stored is sent to a specific chip and die by traversing the channel. At this point a shared register is used to buffer the data and then said data will be written into a specific page inside one of planes on the

die. Read operations occur in the opposite manner with a page's data being sent to the die register and then sent out via the channel to the controller.

Read and write operations occur at the page granularity, while erase operations occur at the block granularity [3]. The difference in the scope of these operations is due to the fact that erasing a single page of a block, given the structure of NAND flash, would cause a disturbance to all other pages within the block, thus leaving the data unreliable [5]. Reads are the shortest operation with a duration of roughly tens of microseconds [26]. Writes take roughly hundreds of microseconds to one or 2 milliseconds, while the erase operation has the longest latency on the order of several milliseconds [25, 26]. The read operation's speed is heavily dependent on the number of bits stored per cell. The larger the number of bits in the cell the more reads to a wordline are required to determine the state of each cell. Speeds of the write and erase operations, on the other hand, are influenced by the size of the page and the number of pages in the block respectively. As size increases, so does the latency. The relationship between size and latency is attributed to the idea that the operation is only as fast as the slowest cell [2].

2.1.3 Out-of-Place Write Policy

When data in a page needs to be updated, a direct overwrite of the page cannot be performed as NAND flash requires that the page must first be erased. However, if an erase occurs at the block granularity, all other pages within the block are also erased causing data loss. For this reason, SSDs invoke an out-of-place write policy [11]. When a page's data is updated, the Flash Translation Layer remaps the logical page of the old data to a new physical page while the original page is marked as invalid leaving two copies: 1) a valid page that holds the new data and 2) an invalid page that holds the out dated data. Figure 2 depicts this write/update policy.

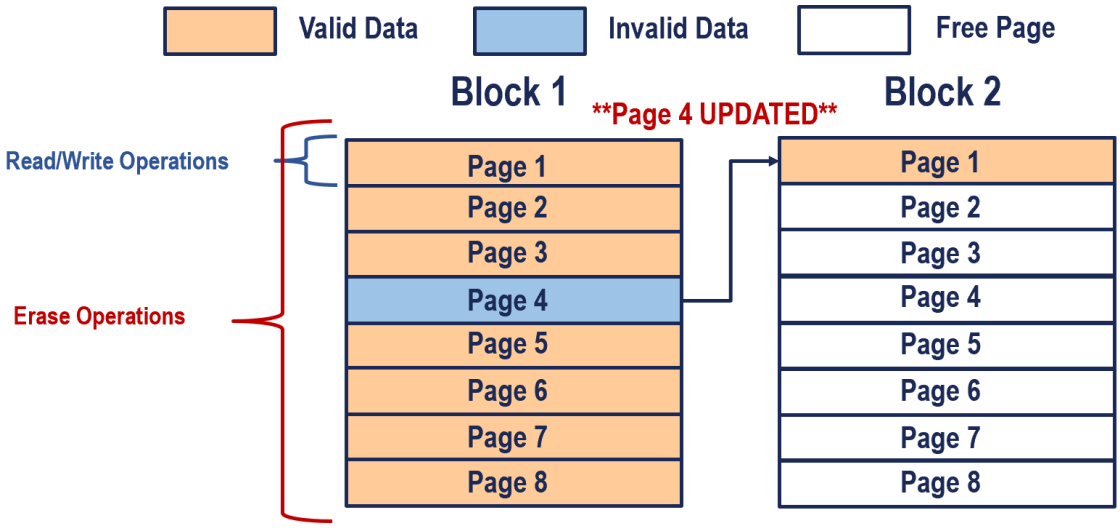

Figure 2: Basic Operations and Out-of-Place Write Policy

2.1.4 Garbage Collection (GC)

As the SSD accumulates more data it eventually needs to reclaim the space being taken up by the invalid pages. Garbage Collection (GC) is the operation invoked to 1) select a block of data to reclaim and 2) there after copy all remaining valid pages in the block to a new physical location as well as 3) perform the erase operation.

GC is triggered when the number of free pages within a plane drops below a given threshold. As per the previous statement, when a block is selected for GC it may still contain some valid pages. In order to ensure these pages are not lost, the valid data is first copied to another block in a different part of the SSD. After all copies are created, the previously valid pages are marked invalid, along with the rest of the block, allowing the erase operation to take place. Unfortunately, the copying of valid pages during a GC operation has an adverse effect on

the lifetime of the SSD as it depletes the already limited number of program/erase (P/E) cycles that the cells of a page can endure. This limitation can be attributed to the deteriorating oxide layer of the cell in addition to trapped electrons that eventually render the cell, and consequently, the wordline unreliable and unusable [18]. Since the writes that occur during GC are not writing new data, these writes are viewed as extra writes that would not occur if GC was not invoked. The phenomenon of additional writes as a result of GC is known as *write amplification* [16]. Write amplification can be numerically calculated by simply subtracting the number of write requests to the SSD from the number of actual writes that occurred internally within the SSD.

2.1.5 Contention of Channels and Dies

When GC takes place in a plane, it stalls incoming I/O requests to itself and other planes in the die [3]. This stalling happens because the planes of the die share a common register. (1) As the contents of the valid pages are being copied out, the data must first be read from the page to the die register. (2) Then the data is transferred out via the channel to the channel controller where ECC is used to correct any errors [4]. During this transfer time the channel is also blocked. (3) Next, the data is sent to the die register of its new physical address and (4) then written to a clean page. Figure 3 illustrates the process of moving valid pages during GC.

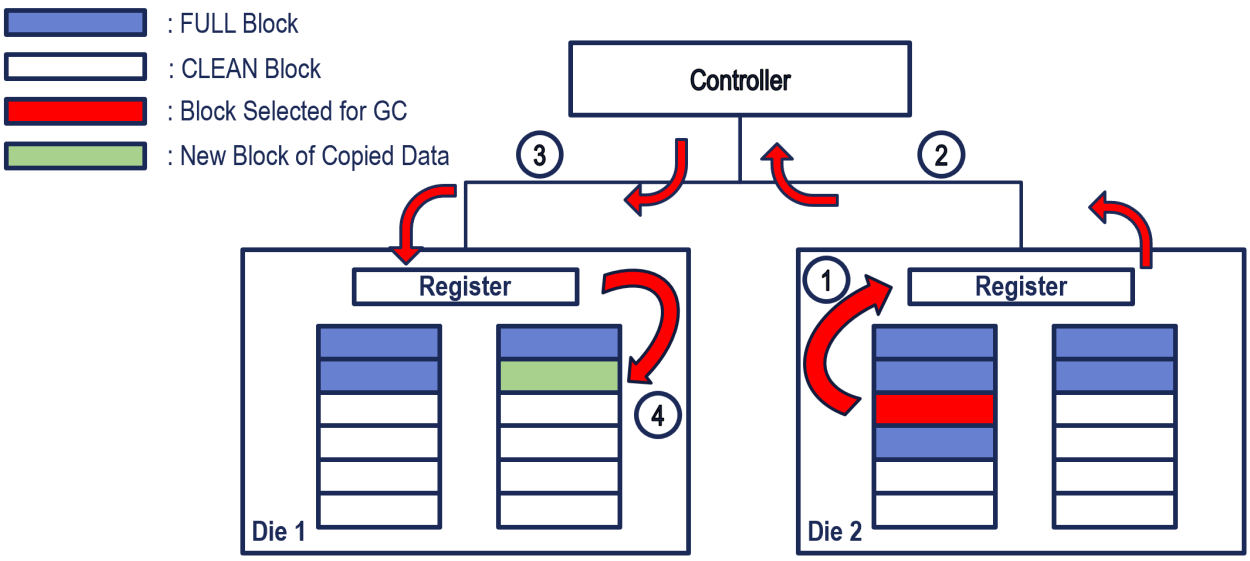

Figure 3: Flow of Valid Pages During Garbage Collection

The blocking of the channels and dies delay other requests from being serviced along these paths. The erase operation is also an issue as it is the longest single operation on the order of several milliseconds and its latency continues to increase in newer generations of SSDs with larger page and block sizes being introduced [4, 25]. Figure 4 highlights the points of contention that GC has with incoming I/O requests.

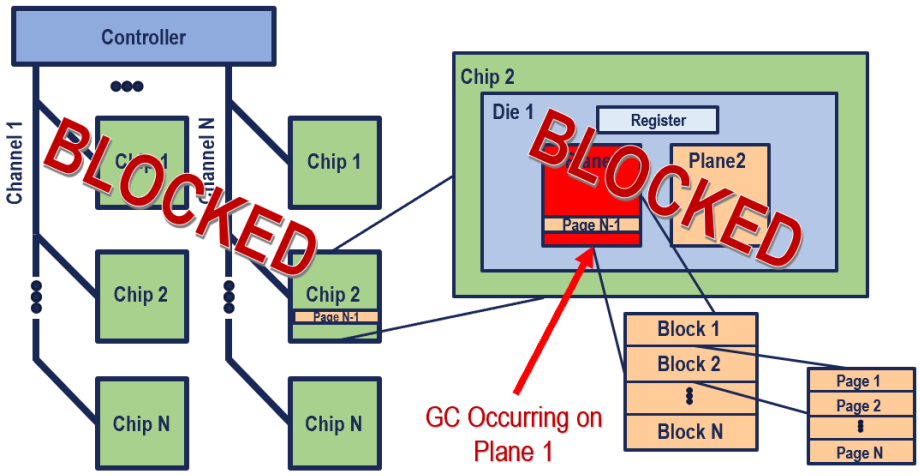

Figure 4: Points of Contention During Garbage Collection

2.2 ADVANCED COMMANDS

SSDs rely on the intrinsic parallelism that comes from their architecture, as previously shown. At the most basic level, parallelism is achieved by sending requests from the controller to several chips on separate channels. Inside each individual chip parallelism becomes slightly more complicated. Fundamentally, when a request/command arrives at the chip its aim is to extract/insert information from/to a single page. In order to find this page, the command needs to address which die, plane, and block it resides in. However, there are many times when the information being requested/provided by the SSD, for the system, is larger than a single page requires multiple operations to the block to serially read/write the data. Intra-chip serial operations from/to a particular location start to become insufficient in this case. To further improve performance, advance commands have been introduced and are used to parallelize operations at the die and plane levels. Figure 5 illustrates the two types of commands available:

1) Multi-Die and 2) Multi-Plane. [6] The Multi-Plane command (red) allows the same operation to occur on two planes within a single die with the condition that they operate on the same block and page offset. The Multi-Die command (white) expands this further by issuing the same command across different planes on different dies with the same stipulations as Multi-Plane. Multi-Die is also capable of issuing the same Multi-Plane command on separate dies of a chip. These advanced commands can be fully taken advantage of when coupled with data striping so information that is likely to be needed together can be written to the same offsets across the chip and extracted at the same time in parallel [15, 27]. Currently, no command provides parallelism below the plane level. This is primarily due to the limitations of peripheral circuits as well as a shared die register that places limits on the volume of data leaving from a single plane at once.

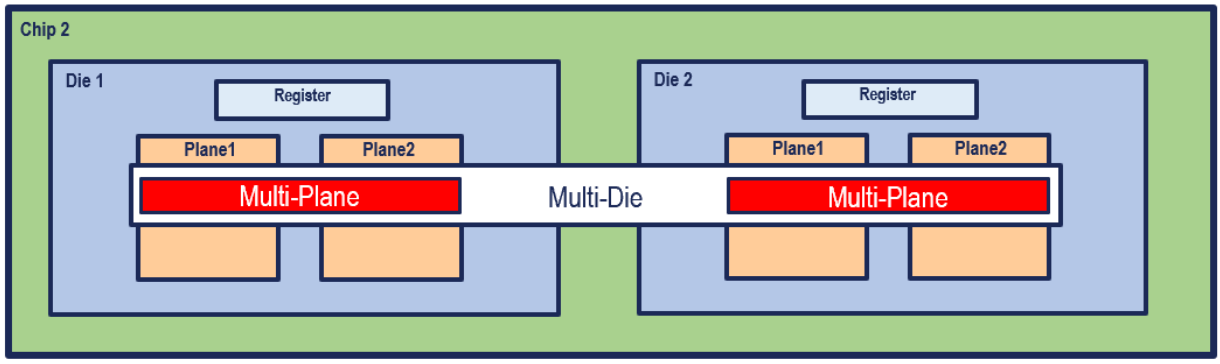

Figure 5: Multi-Plane and Multi-Die Advanced Commands within a NAND Flash Chip

3.0 MOTIVATION

The copying of valid pages and the latency of the erase operation during GC impose a performance and endurance burden on the SSD. As the demand for SSDs to have more space continues, more bits are being stored per cell. However, by doing so there is a decrease in the lifetime of each page causing the number of tolerable writes prior to wearing out to decrease significantly. The extra writes that occur during GC can accelerate this degradation process and decrease the overall lifetime of the SSD. Additionally, contention of the channels and dies during each phase of GC can cause major slowdowns to the system by blocking incoming I/O requests and therefore diminishing quality of service to the user. Thus, it has become paramount that the adverse effects of GC's page movements and erase operations are limited and GC maximizes its efficiency to reclaim space when invocation is necessary.

3.1 RELATED WORK AND CORRESPONDING LIMITATIONS

Prior works have attempted to address the issues derived from GC in different ways and can be generally summarized into the following three categories: 1) Decrease the latency of GC [5, 28], 2) Decrease the frequency of/delay GC [3, 7], and 3) Try to circumvent GC when it is happening [4].

In [4] another advanced command is used called Copyback where instead of pages being moved out over the channel and dispatched to a new physical page, they are read to the register and written to a page of a different block within the same plane. This frees up the channel, however, by not sending the data over the channel to the controller the data does not run through ECC at the controller. In addition, [4] utilizes RAIN to build in redundancy so that in the event a request is blocked by a plane performing GC the parity can be used to generate the blocked data. Unfortunately, the need for redundancy requires additional capacity.

[5] uses a different approach to try and decrease the latency of GC by erasing portions of a block called sub-blocks. These sub-blocks require some pages (or wordlines) to serve as buffers to prevent the erasing of sub-blocks from disturbing data in the other sub-blocks of the block. Pages serving as buffers cannot hold any data and therefore reduce the capacity of the SSD. Dividing the block up in this way means it is easier to find a section to reclaim with few valid pages to move. However, it was found that the erase latency of erasing a single sub-block is essentially equal to that of the latency to erase the entire block. This is due to the fact that the erasing process does not change by having less pages. Therefore, less space may be cleared given the same amount of time resulting in a less effective GC operation.

As mentioned prior, when a plane goes through the GC procedure it occupies the register when copying pages, thus blocking the other plane on the die from servicing incoming requests. [3] decides to make use of this idle time that the other plane experiences by running GC on both by using the Multi-Plane advanced command to move pages and erase a block in each plane of the die in parallel. The idea is that performing GC early in the other plane will delay the need for GC in the future. Although, this method has its limitations. Due to the nature of the Multi-Plane command these operations must share the same block address and page index. For the read and

write commands the pages being read or written to must share an index, otherwise they will be moved one at a time, serially. Additionally, the plane that needs to reclaim space will select the candidate block with the least number of valid pages in order to copy less pages. However, the Multi-Plane command requires that the block in the other plane be the same block index and this block may have many valid pages to move, worsening the latency of GC. Endurance may also be harmed as the adjacent plane may not need to reclaim space yet and therefore the page movements and erasing are unnecessarily speeding wearing.

3.2 EXTENSION OF PARALLELISM

The GC problem is not one that exists equally at all points in the lifetime of the SSD. It is a by-product of SSD aging and more data being stored/updated overtime. The issue worsens as blocks and pages wear out and are replaced with a finite number of built-in spares. In particular, once GC begins to happen it will occur much more frequently. Therefore, GC becomes an unavoidable problem that must be addressed. One of the hallmarks that make SSDs a more attractive storage device than HDDs is the ability to have equal access time to any page of data inside the NAND array. While this may be true, it does not consider delays resulting from contention of channels and dies, both of which are amplified during GC and delay access times.

Contrary to HDDs, which perform better when their data is not fragmented, SSDs perform better when their data is separated. Specifically, if the data pertaining to a file is striped across chips on different channels performance can be improved as each section can be retrieved/written in parallel. Pages in SSDs are actually considered fragmented, in the traditional sense, when the information of a file is written into the same chip, die, or plane [15]. However,

while striping can improve performance it also creates additional points of possible delay. If GC is blocking any point along the path of the striped data then the file cannot be fully retrieved or written until GC is finished.

For this reason, it is desirable to speed up the GC process and improve its effectiveness. Since GC is based on a threshold of how many free pages remain in a plane, the GC process fights to get back over the acceptable threshold. However, if reclaiming one block is not enough then it must continue to serially erase blocks and moving their corresponding valid data until enough space has been reclaimed. In this case, one GC operation was not efficient enough at reclaiming space. Unfortunately, GC and its associated operations are a level below that of the lowest level of parallelism SSDs can currently achieve, even with advanced commands.

Motivated by these issues, this thesis aims to bring parallelism a step deeper into the block-level of the NAND flash hierarchy in order to speed up GC when it is inevitably needed. Additionally, this thesis aims to maximize GC efficiency, with respect to space reclamation, in hopes to limit the number of erase operations and page movements that harm endurance and expel energy. By introducing a new advanced command called Intra-Plane Parallel Block Erase these goals can be achieved delaying the need for GC and minimizing its adverse effects.

4.0 IPPBE: INTRA-PLANE PARALLEL BLOCK ERASE

Since SSDs have become a prominent storage media, it is advantageous to find solutions that do not impose drastic changes to the internal architecture that may require changes to the infrastructure of the system. With previous designs in mind, this thesis seeks to address GC by leveraging the NAND flash structure in its present design without making sacrifices to capacity and maximize the efficiency of each GC. Intra-Plane Parallel Block Erase, a new advanced command, is able to meet these requirements as well as be combined with existing designs to further enhance them.

4.1 DESIGN OVERVIEW

Intra-Plane Parallel Block Erase, or IPPBE as it will be referred to throughout the rest of this thesis, is, to the best of the author's knowledge, the first advanced command to parallelize at the block-level within the same plane. Its design is based on exploiting the triple-well structure shared by all blocks residing in a plane [2, 11]. The well plays an important role during the erase operation. As the well is biased with a high voltage, the wordlines of the block being erased are grounded to allow the electrons stored in each cell to be flushed out. The blocks that are not being erased float their wordlines. Blocks are selected via a block decoder in the plane. By redesigning the decoder to select multiple blocks at a time, multiple blocks can be grounded

during the erase operation and therefore be erased simultaneously. Erasing more than one block at a time increases the space reclaimed during an erase operation without increasing the erase latency. In addition, future GCs will be delayed due to more space being reclaimed per GC and natural cold data clustering that occurs as a result of IPPBEs implementation. IPPBE can also accelerate the process of getting a plane's free page count back over the GC barrier so that it can resume servicing blocked reads and writes sooner. Figure 6 visualizes IPPBE's implementation.

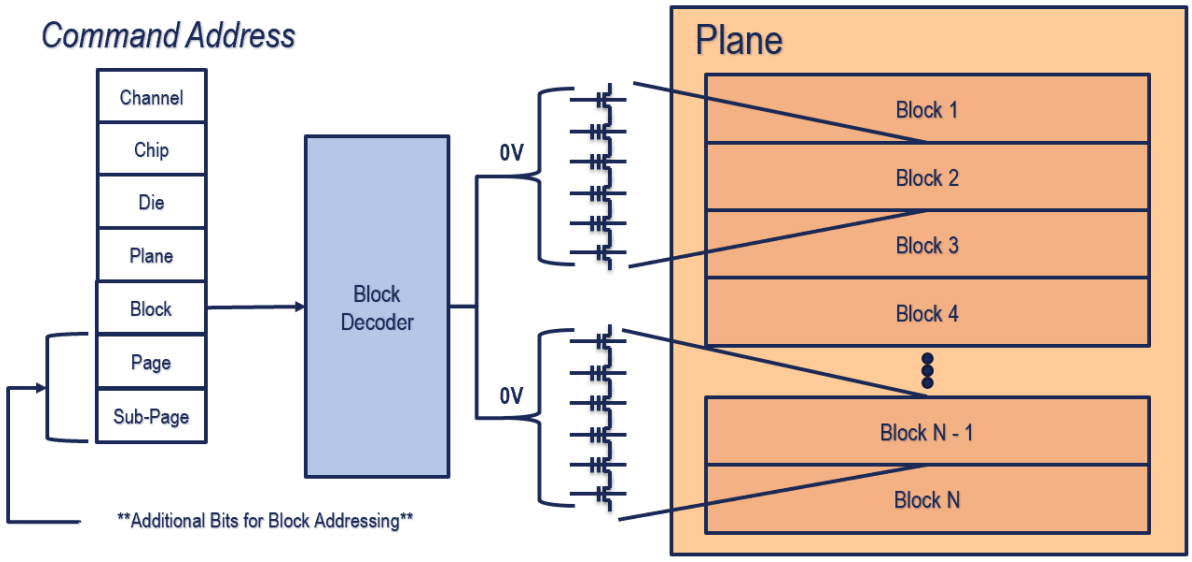

Figure 6: Intra-Plane Parallel Block Erase (IPPBE) Design

4.2 ERASE OPERATION

To understand IPPBE's design it is important to understand the erase operation's protocol. [5] found that erasing portions of a block did not reduce the latency of the erase operation. This

makes sense as the steps taken during the erase operation procedure do not change regardless of the number of pages in a block. [11] describes this process in detail and Figure 7 provides a visual for the steps taken to erase a block. After all the data has been copied out, the erasing can begin. First, all the wordlines of the selected block are programmed to get each cell into the same state. Next, the wordlines of the selected block are grounded and the remaining blocks are left floating. From this point the process enters a loop of electrical pulses and verification. The iP-well, common to all blocks in the plane, receives an electrical pulse of V_Erase. After which, all worldlines are read to verify that the cells have been erased and read as 1. If not, V_Erase is increased by V_step and the electrical pulse is once again applied to the iP-well. This process continues until it can be verified that all cells have been successfully erased. It is normal for erase pulses to be applied several times before an erase is successful. It should be noted that there is a limit to the number of times this loop can run before the erase is deemed a failure. This procedure is the same regardless if the number of pages per block is 64 or 256. Rather than reducing the size of the erase unit to a portion of a block, IPPBE does the opposite by adding more blocks. By doing so, multiple blocks can be reclaimed in the same amount of time it takes to erase a single block, maximizing the erase operation.

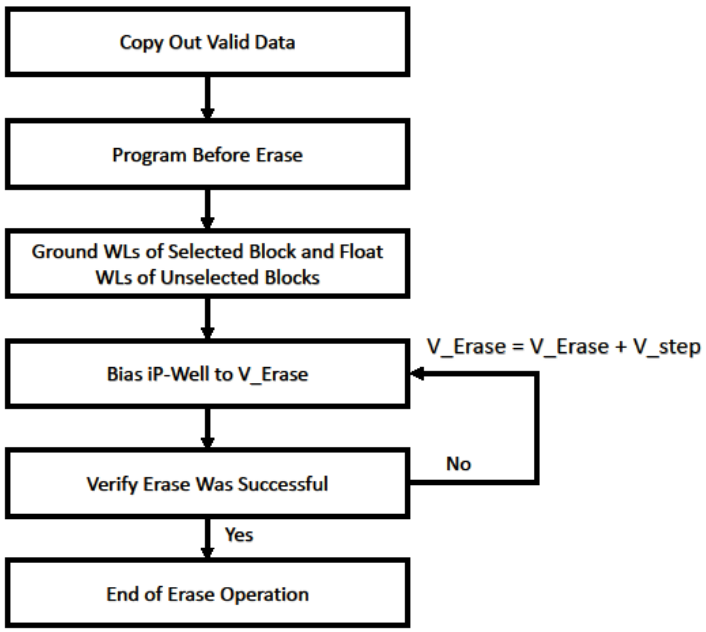

Figure 7: Erase Operation Procedure

4.3 NAND FLASH PLANE TRIPLE-WELL

In this design the triple-well structure is not altered, however, it is the key component of enabling IPPBE. In NAND flash a triple-well structure is used to construct the plane. There are three sections that the well is divided into: 1) iP-well, 2) N-well, and 3) P-type substrate [2, 9, 11]. As mentioned prior, the iP-well receives the high voltage electrical pulse during the erase operation. When a block being erased has its wordlines grounded during this pulse, the electric potential between the wordlines and the iP-Well induces Fowler-Nordheim tunneling, enabling the trapped electrons in each cell to be flushed out [11]. It is not desirable to have all blocks experience this electron tunneling. Since this iP-well is shared by all blocks in each plane, the other blocks need

to inhibit their wordlines to prevent unexpected erasure and loss of data. To do so, blocks not selected for erasure leave their wordlines floating. This maneuver raises the capacitive coupling so that the electric potential between the wordlines and the well is not sufficient to induce electron tunneling. IPPBE takes advantage of this phenomena. Rather than instructing only one block to ground its wordlines, multiple can be instructed to ground their wordlines during the electric pulse. Since the biasing of the well has a duration that lasts a fixed interval of time, then erasing more than one block does not result in additional time spent performing the operation. An energy analysis was conducted using the FlashPower simulator and shows that for every block added during this erase operation the energy expended grows linearly [12]. However, by using IPPBE, GC is called less frequently and therefore overall energy consumption is decreased. A breakdown of the triple-well structure can be seen in Figure 8.

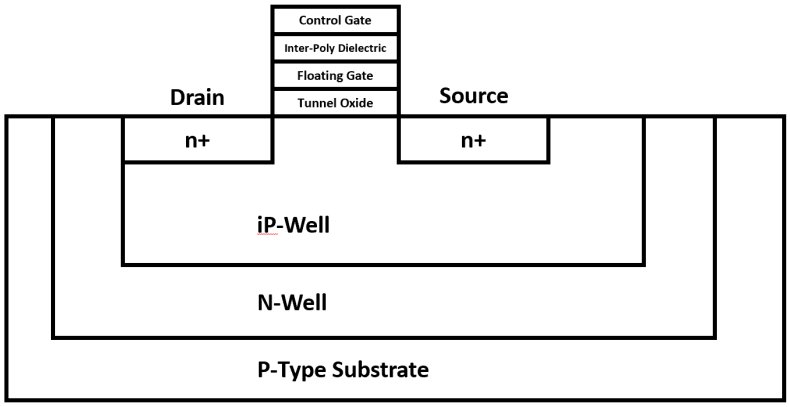

Figure 8: NAND Flash Triple-Well Structure

4.4 DECODER AND COMMAND ADDRESSING

When the SSD decides which block will be erased a block decoder is used to set the wordlines. However, given that in IPPBE's design additional blocks are necessary, a slight modification needs to be made to the decoder to allow multiple blocks to be addressed at the same time. [6] notes that the addressing for operations is simply channel, package, chip, die, plane, block, page, and sub-page. Since the erase operation is not concerned with pages or sub-pages these bits could be repurposed to add the address of other blocks being addressed simultaneously. IPPBE addressing could potentially be modified to combine with and enhance previous GC schemes. For instance, IPPBE could be altered to address multiple sub-blocks across different blocks within a plane. In addition, if combined with the Multi-Plane command multiple block erasure could occur simultaneously within a plane as well as in other planes belonging to the same die.

4.5 BLOCK SELECTION

Prior to IPPBE, determining which block is best to reclaim was fairly simple. The most traditional approach is termed Greedy GC, which means the block selected was the one with the largest number of invalid pages [11]. The idea being that by selecting the block with the most invalid pages there would be less valid pages that need to be copied out. This way the most space possible is reclaimed when erased. However, with IPPBE careful attention should be made to selecting which blocks to reclaim. For simplicity, consider selecting two blocks to erase. The plane free page count is reduced to the point of triggering the need for GC. Within the plane, two blocks have all invalid pages except for two valid pages each. IPPBE can erase both blocks after

all four valid pages are moved out prior. In this case a few extra pages are moved initially to reclaim twice as much space. This will delay the need for GC longer than erasing only one. However, there is also a scenario where GC is triggered and there are many valid pages in the plane. In this case a block that is entirely invalid may be selected, however, the block with the second least number of valid pages only has 60 percent of its pages invalid. This means that, assuming no free pages in the block, 40 percent of the pages must be copied out. In this case it is better to only erase one block. If both were selected to be reclaimed, the time spent moving all the valid pages would counter-act the benefit of simultaneously erasing blocks.

To solve this problem, thresholds are introduced to limit the number of pages being copied. The SSD first finds the best block to erase similar to Greedy. Then, it finds the second-best block. Next, an analysis is done to determine if the number of pages that would need to be copied are below the threshold. If yes, then both can be reclaimed. If not, then the second block is not selected to be reclaimed.

4.6 GARBAGE COLLECTION BARRIER

One of the issues with GC is that once it starts to occur it typically will continue to happen. The trigger is usually the passing of a threshold that brings the number of free pages below a given percentage of the total number of pages in the plane. Once this happens, space must be reclaimed continuously until the number of free pages has crossed back over the threshold, or Garbage Collection Barrier (GCB). However, once enough space is reclaimed the plane can return to servicing read and write requests. Because not enough time may be provided to claim more space, there may be thrashing between the plane reclaiming space and writing to the plane.

IPPBE helps to prolong this need to return to GC by reclaiming more space and further distancing the number of free pages from the GCB. Figure 9 illustrates this concept. On the other hand, there may be a large write that pushes the free page count far past the barrier. IPPBE can accelerate the process of returning to the other side of the GCB by reducing the number of erase operations it takes to get there.

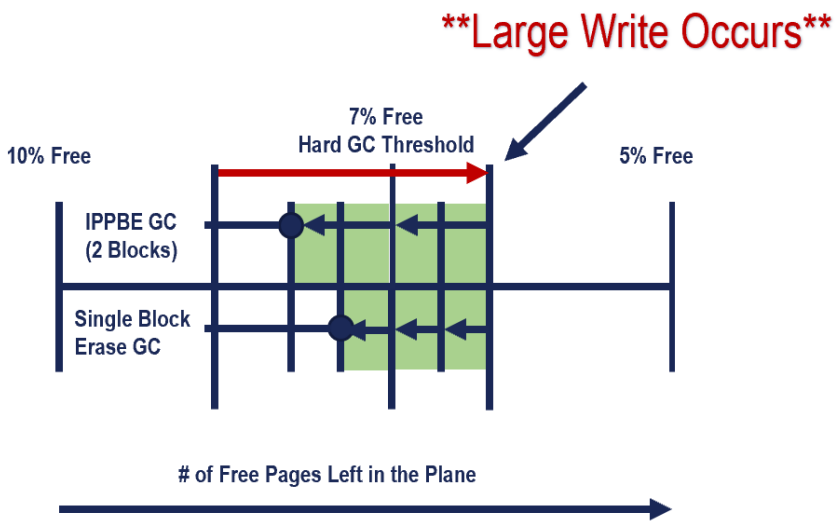

Figure 9: IPPBE vs Single Block Erase with GC Triggered at 7 Percent Free Page Count

4.7 COLD DATA CLUSTERING

A frequently considered topic in SSDs concerns rating data based on how frequently it is updated/written and henceforth requires that a new page be written. When a page update occurs,

a trail of invalid pages is left behind consuming capacity due to the out-of-place write policy. Prior works have attempted to classify data into one of two categories: hot or cold [19]. Hot data being data that is frequently updated, while cold data is written and updated very infrequently, if at all. Another way to describe this is through intensity of reads or writes. Hot data is write intensive and cold data is read intensive. The central focus of these works is to place hot data together within a block and cold data together within its own block [19].

The theory behind this concept is rooted in the GC procedure. When hot and cold data are mixed together, GC's negative effects are often the worst. The reason being is when pages in a block are a mix of hot and cold, the hot data will likely turn into an invalid page while the data that remains valid in a block, when GC selects it, is often cold. If every block of the plane contains a relatively even ratio of hot to cold data then GC is forced to select a block that needs to move a large number of valid pages. However, if the data can be separated, a hot block will a produce a block of mostly invalid pages and thus a good candidate for GC to reclaim. The cold blocks will not be selected and won't have many invalid pages so their effect on capacity is minimal. Unfortunately, classifying data this way is very difficult as data read/write intensities are rather unpredictable over time.

IPPBE is able to naturally cluster code data into its own block and overtime reduces the overall latency of GC and frequency of GC. Traditionally when writing to a plane in SSDs and NAND flash, a block is selected as the active block. When the active block fills up, the next free block becomes the active block. This happens in a round robin fashion for wear leveling purposes. When triggered, GC will copy valid pages of the victim block to the active block, recall that these pages are often cold data. Since GC only erases one block at a time traditionally, other write operations may fill the block in between GC invocations thus creating a mix of hot

and cold data again. IPPBE works differently. Since multiple blocks are being selected to be reclaimed simultaneously, there may initially be more pages to move. However, since this new GC procedure is an atomic operation GC is able to fill the active block with a larger ratio of cold data without new writes of hot data arriving at the active block between erase operations. This natural clustering of cold data from multiple locations means that in future GCs the probability of finding hot blocks of mostly invalid pages increases. Reclaiming blocks in this manner means that blocks can be reclaimed with fewer valid page copies and at the same time reclaim more space per GC operations. Overtime, this clustering of cold data results in the contention being lowered as GC will create less traffic. Additionally, each GC will be shorter due to the fact that less pages need moved. Finally, more space is reclaimed per erase operations and therefore GC is triggered less frequently. Figure 10 demonstrates how cold data left over in the victim blocks of GC are cluster together in the active block of the plane when using IPPBE.

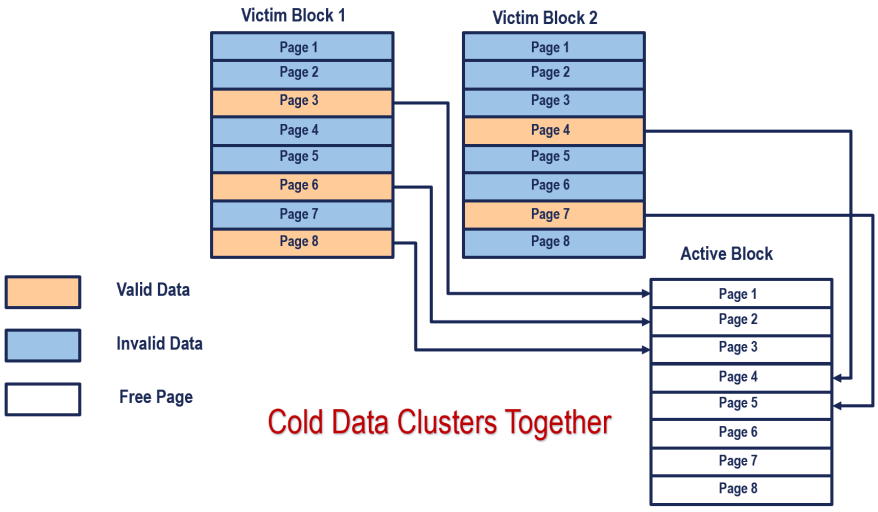

Figure 10: IPPBE Cold Data Clustering in the Active Block

5.0 EVALUATION OF RESULTS

5.1 EXPERIMENTAL METHODOLOGY

5.1.1 Simulation Environment and Configuration

IPPBE performance was evaluated using a combination of two different simulators SSDSim [6] and FlashPower [12]. SSDSim is an event-based simulator that uses traces [13] of I/O requests to track read and write operations through a desired SSD configuration. The configurations for the evaluation of this paper can be seen in Table 1. SSDSim's GC algorithm was modified to implement IPPBE's design by introducing additional mechanisms to find multiple candidate blocks to reclaim. Timing schemes also received alterations to ensure that erasing multiple blocks only occupies the channel and chip the same way as erasing a single block. For the purpose of this work the maximum number of blocks erased at a time was limited to two. A static allocation scheme was implemented, prioritizing in the following order: channel, chip (also referred to in some works as way), die, and plane with channel being highest and plane being lowest. FlashPower was used to evaluate the energy consumption of the design. Given the same configuration as in Table 1 the simulation can output the energy consumption of the read, write, and erase operations. Both of these simulators have been validated against the real device.

Table 1: SSD Configuration

SSD Configuration			
Channels:	**2**	**Read Latency:**	**75us**
Chips Per Channel:	**2**	**Write Latency:**	**1.5ms**
Dies Per Channel:	**2**	**Erase Latency:**	**3.8ms**
Planes Per Die:	**2**	**Channel Transfer:**	**25ns/byte**
Blocks Per Plane:	**2048**	**Overprovision:**	**25%**
Pages Per Block:	**64**	**GC Threshold:**	**7%**
Page Size:	**2KB**		

5.1.2 Inducing Garbage Collection

Since not every trace is the same size the SSD needs to be aged to force GC to occur. SSDSim does this by injecting invalid pages into the SSD. However, this aging process has been found to not be consistent with how SSDs truly age [8]. SSDSim ages the SSD by filling each plane with invalid pages to a percentage provided by the user. For instance, if the user aged the SSD to 75% and each block contained 128 pages, then the simulator preprocesses all blocks to the have the first 96 pages invalid and the remaining 32 free. In reality, the SSD typically writes to blocks residing in the plane via a round robin fashion due to wear leveling. Thus, it is not likely that the free pages of an SSD would be distributed in this manner. Typically, there is a single active block within the plane and if data is directed to that plane it will first fill up the active block and proceed to the next one once the active block is full. Additionally, the aging process should have

some valid data mixed among the invalid pages otherwise the entire SSD would be viewed as logically free which is highly unlikely. This strategy for aging would not incur the valid page movements required to be analyzed in this study. Instead blocks would simply be erased which effectively renders the aging impractical in this study. To solve this issue, the configuration used is relatively small. This allows GC to occur naturally rather than being artificially imposed. GC was set to trigger when the number of free pages in a plane dropped below 7%. Figure 11 demonstrates the difference between SSDSim aging and aging of real SSD. In this instance aging was set to 75% of total capacity.

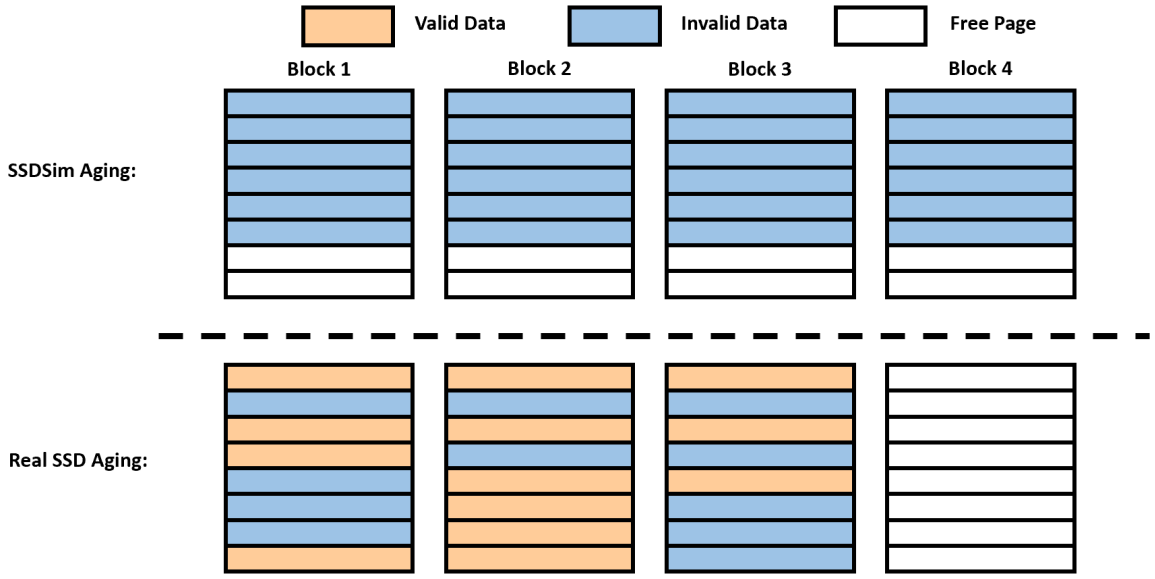

Figure 11: SSDSim Aging (Top) vs Real SSD Aging (Bottom). Both Aged to 75%

5.2 RESULTS

Evaluation of IPPBE's design is divided into endurance, performance, and energy. For comparative purposes Greedy GC is used as a baseline for which IPPBE is normalized against. Each scheme was evaluated using a suite of traces from SNIA IOTTA repository [13]. Table 2 provides statistics on the traces used.

Table 2: Trace Statistics

System Traces			
Trace Name	Write Ratio	Read Ratio	Total Requests
hm0	64.4%	35.6%	3,993,316
prn0	89.2%	10.8%	5,585,886
prxy0	96.9%	3.1%	12,518,968
proj0	87.5%	12.5%	4,224,524
src20	88.6%	11.4%	1,557,814
stg0	84.8%	15.2%	2,030,915
ts0	82.4%	17.6%	1,801,734
src12	74.6%	25.4%	1,907,773

5.2.1 SSD Lifetime and Endurance

The SSDs lifetime is a growing concern especially given that SSDs using MLC or TLC designs to hold multiple bits per cell have a significant reduction in the lifetime of pages due to accelerated wear-out and tighter threshold voltage distributions [1, 30]. GC worsens this problem by requiring pages be moved thus inducing extra writes. IPPBE tries to limit this by reducing the number of pages movements and block erases. Figure 12 and Figure 13 compare the total number of pages moved and blocks erased during GC. Experimentation shows that the number of page movements are reduced by up to 52.2% and 26.6% on average and the reduction in the number of blocks erased in some traces can be up to 14.2% and 3.6% on average. When IPPBE is applied, both blocks move their pages together then perform the erase. Since no operations occur in between moving pages of the different blocks when using IPPBE, the pages are likely written to the same active block. Data in pages that are being moved are most likely read intensive data since they are among the last few pages yet to invalidate in the block. These frequently read pages are less likely to be updated and may need moved in a later GC. The more fragmented the valid pages are across the plane, the likelihood of needing to copy data during the next GC increases. IPPBE naturally clusters this type of data into the same block. As a result, the next GC on the plane moves less pages because there is a higher chance of being able to select a block with less valid pages. The traces able to reduce page copying, due to clustering of valid pages, saw a decrease in blocks erase because IPPBE was able to reclaim more blocks with a larger number of invalid pages, thus increasing the amount of space reclaimed per erase. This increase in erase efficiency can accumulate overtime leading to less blocks needing to be erased because enough space has already been reclaimed. This is more apparent in larger and longer

traces such as prxy0 and prn0. This reduction in page copies and block erasures helps to alleviate the endurance burden imposed by GC.

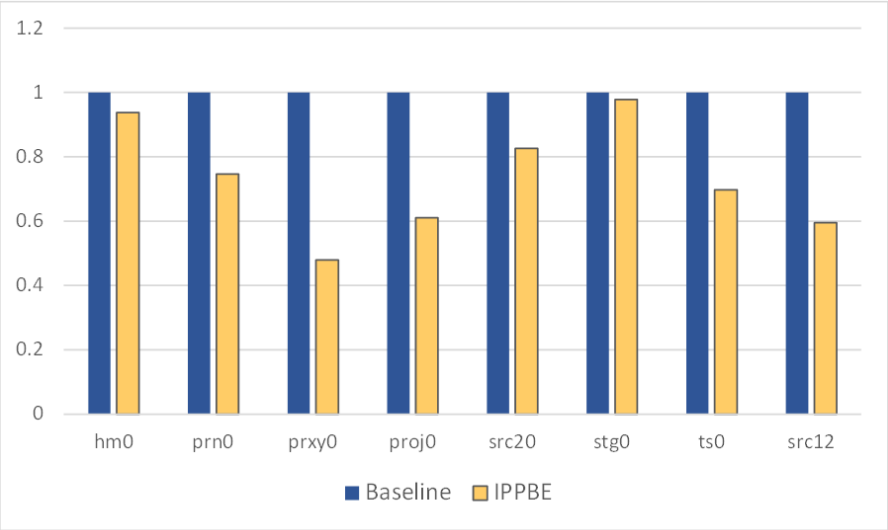

Figure 12: Valid Pages Moved During Garbage Collection

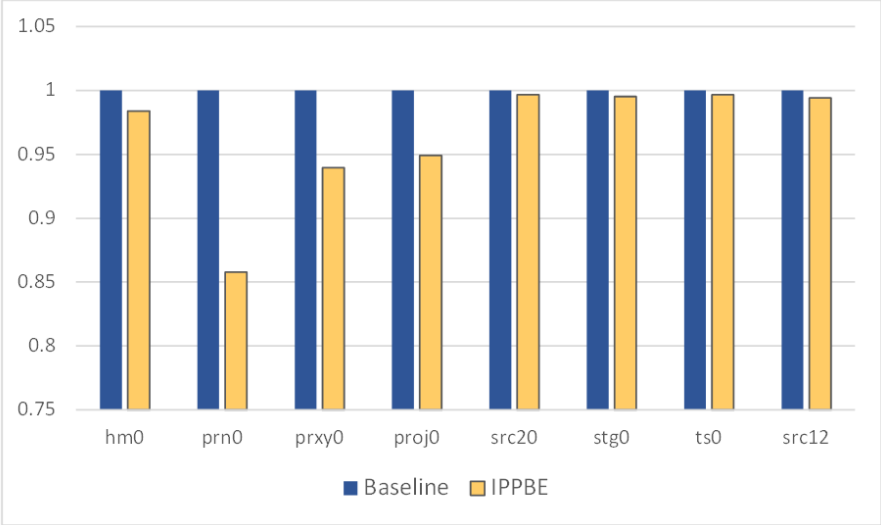

Figure 13: Total Blocks Erased

5.2.2 Read/Write Performance

GC is detrimental to performance of SSDs because when required it takes priority over all other requests to the die. Therefore, it is advantageous to limit the time spent reclaiming space. Figure 14 demonstrates that IPPBE can reduce the total time spent performing GC by up to 50.7% and 33.6% on average. The reduction is very closely related to the number of times the GC mechanism is invoked. Figure 15 helps show the relationship between the number of times GC is triggered and the time spent on GC. GC frequency is reduced up to 56.4% and 46.7% on average. The reason being is IPPBE can more quickly get back to the other side of the GCB by reclaiming more space per unit of time. However, reducing the frequency of GC operations does not always result in a linear reduction in time spent on GC. The reduction in time spent can be attributed to its heavy reliance on the ability to reduce the number of valid page copies. For

instance, stg0's reduction in page copies is the smallest and in turn sees the smallest reduction in time spent on GC. This is because the cumulative latency of moving pages is often worse than that of the erase operation. Nevertheless, being able to efficiently erase and move pages leads to a long-term reduction. While it is possible the initial use of IPPBE could see a longer GC latency due to page movements, this eventually leads to high efficiency of GC thus limiting its need and time spent performing the operations associated. Figure 16 and Figure 17 demonstrate that reducing the time spent on GC has a ripple effect on the overall average read/write response times of the trace. The average read response time is reduced up to 47.0% and 16.5% on average. The average write response time also decreases by up to 45.4% and 14.8% on average. The results vary based on how large of a portion the baseline's time is spent on GC.

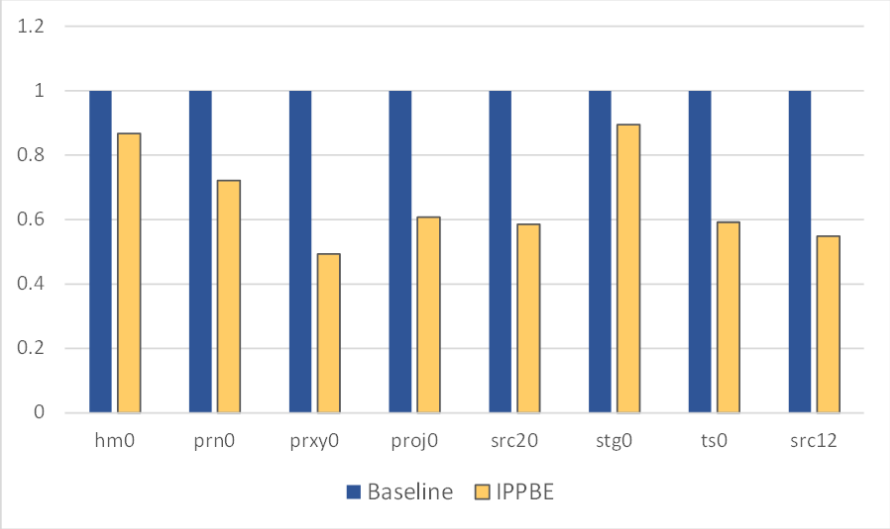

Figure 14: Total Time Spent on Garbage Collection

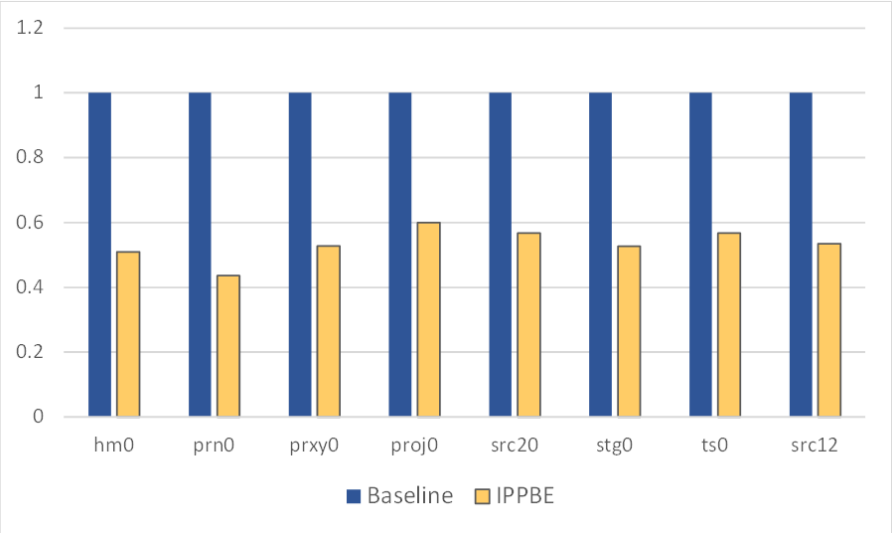

Figure 15: Garbage Collection Invocations

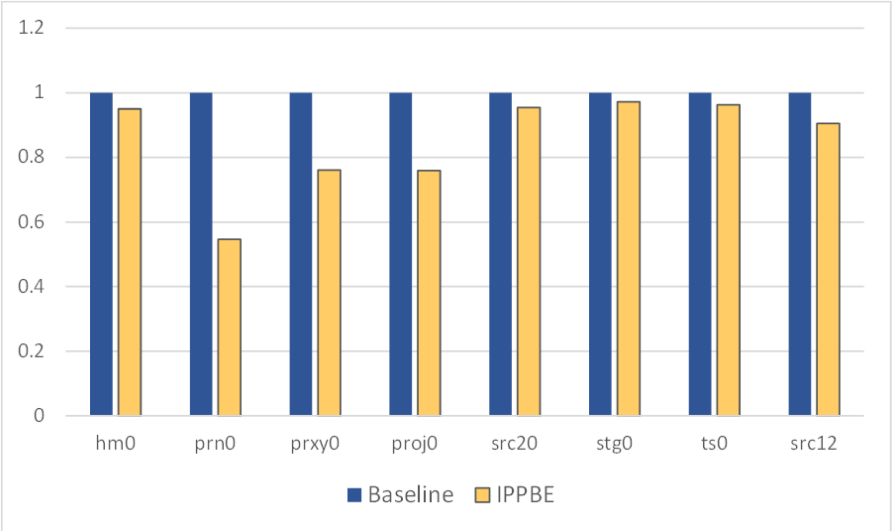

Figure 16: Average Write Response Times

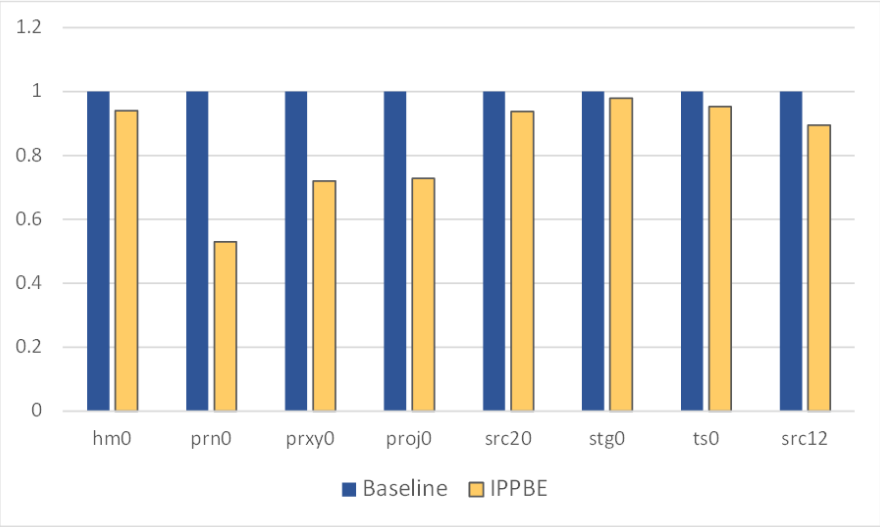

Figure 17: Average Read Response Times

5.2.3 Energy Analysis

An energy analysis was conducted to investigate the energy impact of using the IPPBE command. The FlashPower simulator was used to calculate the energy consumption of the read, write, and erase operations given the SSD configuration provided in Table 1. The results indicated the following energy consumptions: Read = 2.1uJ; Write = 12.3uJ; and Erase = 22.5uJ. Calculation of energy consumption for GC was as follows: Each page copy requires a read and write command so the energy to move pages during GC is the sum of the read and write energy multiplied by the number of pages moved. Then, this value is summed with the product of the erase energy and number of blocks erased. Using FlashPower and [12] it can be concluded that the energy consumption of adding an additional block during the erase pulse causes the erase energy to increase linearly as the majority of the energy is consumed by the bitlines. Figure 18

shows that the savings in energy expended by GC can be as high as 44.1% and 19.3% on average. The results are a by-product of the reduction of pages moved and blocks erased in each trace. The GC energy savings most closely resembles the results of the pages movements because there tends to be significantly more pages moved than blocks erase, therefore the write energy becomes the dominate contributor. It should be noted that these values change based on the size of the plane.

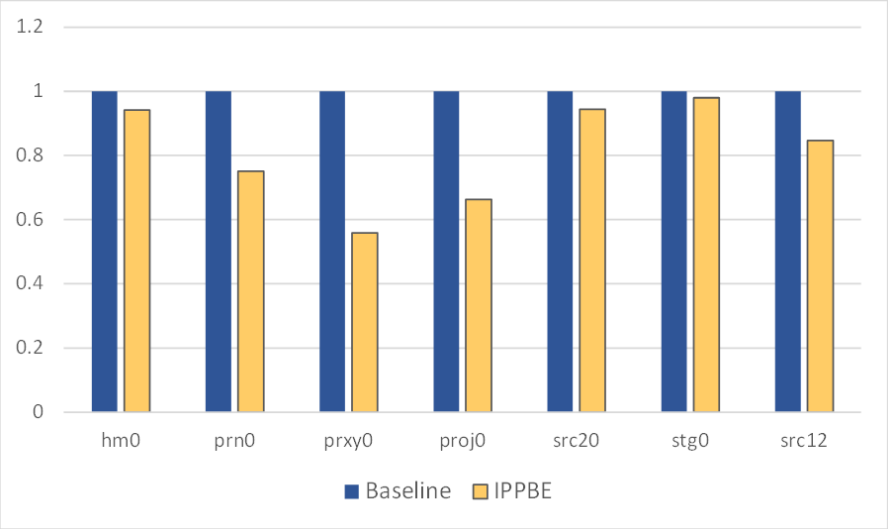

Figure 18: Garbage Collection Energy Expenditure

6.0 FUTURE DIRECTION

6.1 COMBINING IPPBE WITH PRIOR DESIGNS

As previously mentioned, IPPBE is an orthogonal design that has the potential to work in conjunction with previously proposed design schemes and already existing advanced commands. Figure 19 demonstrates an example of how IPPBE can be coupled with the sub-block design and the Multi-Plane advanced command. As shown below, IPPBE can be modified to select multiple sub-blocks within a plane which would increase GC efficiency of space reclamation by moving a fewer valid pages than larger blocks prior to the erase operation. As a result, by erasing multiple sub-blocks simultaneously and with careful selection of victim sub-blocks, the ratio of invalid pages to total pages being erased approaches one. A ratio of one signifies as scenario where all pages within the scope of the erase operation are invalid and therefore no page copies are necessary. Additionally, the Multi-plane command can be utilized to simultaneously erase sub-blocks on both planes of the die. The result of such command coupling provides a more fine-grained erase operation that can dynamically tune the scope of the erase operation to minimize contention and maximize GC efficiency within a plane as well as the die.

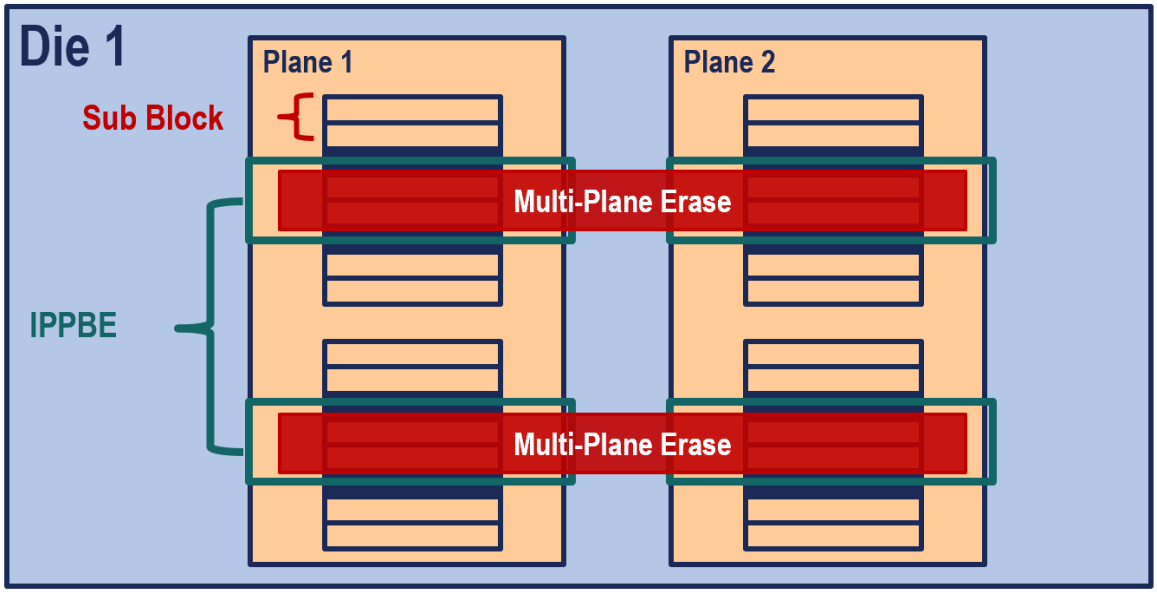

Figure 19: Combining IPPBE with Sub-Blocks and Multi-Plane Erase Advanced Commands

6.2 TRANSITION TO 3D NAND FLASH

It should be noted that this thesis focuses on 2D Planar NAND flash and relies on its structure to achieve the desired results. However, in recent years the demand for increased storage capacity and scaling of NAND flash memories reaching physical limitations, the industry has turned to develop new 3D NAND flash memories [10]. With the advent of 3D NAND flash comes a variety of new physical structures and additional challenges such as disturbance from a third direction and non-uniform programming speeds within a block [31-34]. In turn, in order to bring IPPBE to the 3D NAND flash environment, alterations are likely required to utilize the new structures. Also, given that block and pages sizes continue to grow a scheme such as IPPBE may

require additional limitations and fine grain controls to ensure initial use of the command does not result in abnormally long operations and stalls. In future work, the author desires to extend IPPBE to accommodate these new structural and physical challenges.

7.0 CONCLUSION

To try and mitigate the impact GC has on the system this thesis proposes a new advanced command called Intra-Plane Parallel Block Erase (IPPBE) that leverages the structure of the NAND flash plane to parallelize the erase operations. Intra-Plane Parallel Block Erase (IPPBE) is the first advanced command, to the best of the authors knowledge, to bring parallelism to the NAND flash block-level. By taking advantage of the triple-well structure of the plane and modifying the block decoder, IPPBE can improve endurance, performance, and energy consumption of SSDs and its NAND flash storage medium. As a result of reclaiming more space in the given erase latency, it is possible to reduce the number of times GC is triggered, thus reducing contention and thereby improves the response times of I/O requests. IPPBE alleviates GC's impact overtime by maximizing GC efficiency through the clustering of valid pages (cold data) into the same block to reduce future page copies. Experimentation shows that read/write response times can be reduce by up to 47.0%/45.4% and 16.5%/14.8% on average respectively and the number of page copies by up to 52.2% and 26.6% on average. An analysis of energy consumption was also investigated and concluded that energy used by GC operations can be diminished by up to 44.1% and 19.3% respectively. IPPBE can also be modified to operate in conjunction with previous GC design schemes for further enhancement.

BIBLIOGRAPHY

[1] Y. Cai, S. Ghose, E. F. Haratsch, Y. Luo, and O. Mutlu. Error Characterization, Mitigation, and Recovery in Flash-Memory-Based Solid-State Drives. *Proc. IEEE* 105, 9 (Sept 2017), 1666–1704.

[2] Kang-Deog Suh et al. A 3.3 V 32 Mb NAND flash memory with incremental step pulse programming scheme. *IEEE Journal of Solid-State Circuits* 30, 11 (Nov1995), 149–1156.

[3] N. Shahidi et al. Exploring the Potentials of Parallel Garbage Collection in SSDs for Enterprise Storage Systems. In *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2016.

[4] Shiqin Yan et al. Tiny-Tail Flash: Near-Perfect Elimination of Garbage Collection Tail Latencies in NAND SSDs. *FAST*, 2017.

[5] T. Y. Chen et al. 2016. Enabling sub-blocks erase management to boost the performance of 3D NAND flash memory. In 53nd ACM/EDAC/IEEE *Design Automation Conference (DAC)*, 2016.

[6] Yang Hu, Hong Jiang, Dan Feng, Lei Tian, Hao Luo, and Shuping Zhang. 2011. Performance Impact and Interplay of SSD Parallelism Through Advanced Commands, Allocation Strategy and Data Granularity. *In Proceedings of the International Conference on Supercomputing (ICS),* 2011.

[7] Myoungsoo Jung, Ramya Prabhakar, and Mahmut Taylan Kandemir. 2012. Taking Garbage Collection Overheads off the Critical Path in SSDs. In Proceedings of the *13th International Middleware Conference (Middleware),* 2012.

[8] J Kim, M Park, and I Shin. 2017. Improving SSD Simulator for High Reliability of Performance Evaluation. *International Journal of Applied Engineering* Research 12, 15 (2017), 4836–4839.

[9] Jin-Ki Kim. 2010. Partial block erase architecture for flash memory. (Sept. 28 2010). US Patent 7,804,718.

[10] Rino Micheloni. 2016. 3D Flash Memories (1st ed.). Springer Publishing Company, Incorporated.

[11] R. Micheloni, L. Crippa, A. Marelli, Inside NAND Flash Memories, Amsterdam, The Netherlands:Springer, 2010.

[12] V. Mohan, T. Bunker, L. Grupp, S. Gurumurthi, M. R. Stan, and S. Swanson. 2013. Modeling Power Consumption of NAND Flash Memories Using FlashPower. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 32, 7 (July 2013), 1031–1044.

[13] SNIA IOTTA: Storage Networking Industry Association's Input/Output Traces, Tools, and Analysis. Microsoft production server traces. 2008. http://iotta.snia.org/traces/list/BlockIO

[14] Gala Yadgar, Eitan Yaakobi, and Assaf Schuster. 2015. Write once, get 50% free: saving SSD erase costs using WOM codes. In *Proceedings of the 13th USENIX Conference on File and Storage Technologies (FAST),* 2015.

[15] Sangwook Shane Hahn, Sungjin Lee, Cheng Ji, Li-Pin Chang, Inhyuk Yee, Liang Shi, Chun Jason Xue, and Jihong Kim. Improving file system performance of mobile storage systems using a decoupled defragmenter. In *Proceedings of the 2017 USENIX Conference on Usenix Annual Technical Conference (USENIX ATC)*, 2017.

[16] Xiao-Yu Hu, Evangelos Eleftheriou, Robert Haas, Ilias Iliadis, and Roman Pletka. 2009. Write amplification analysis in flash-based solid state drives. In *Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference (SYSTOR),* 2009.

[17] Feng Chen, Rubao Lee, and Xiaodong Zhang. 2011. Essential roles of exploiting internal parallelism of flash memory based solid state drives in high-speed data processing. In *Proceedings of the 2011 IEEE 17th International Symposium on High Performance Computer Architecture (HPCA)*, 2011.

[18] Xavier Jimenez, David Novo, and Paolo Ienne. 2014. Wear unleveling: improving NAND flash lifetime by balancing page endurance. In *Proceedings of the 12th USENIX conference on File and Storage Technologies (FAST),* 2014

[19] I. Shin, Hot/Cold Clustering for Page Mapping in NAND Flash Memory, *IEEE Transactions on Consumer Electronics*, vol. 57, pp. 1728-1731, Nov 2011.

[20] Y. Cai, S. Ghose, Y. Luo, K. Mai, O. Mutlu and E. F. Haratsch, Vulnerabilities in MLC NAND Flash Memory Programming: Experimental Analysis, Exploits, and Mitigation Techniques, *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2017.

[21] Yu Cai, Yixin Luo, Erich F. Haratsch, Ken Mai, Onur Mutlu, Data retention in MLC NAND flash memory: Characterization optimization and recovery, *High Performance Computer Architecture (HPCA),* 2015.

[22] Kent Smith. Understanding SSD Over Provisioning. In *Flash Summit Proceedings*, 2012.

[23] Bo Mao, Suzhen Wu, and Lide Duan. 2018. Improving the SSD Performance by Exploiting Request Characteristics and Internal Parallelism. Trans. Comp.-Aided Des. Integ. Cir. Sys. 37, 2 (February 2018)

[24] Nitin Agrawal, Vijayan Prabhakaran, Ted Wobber, John D. Davis, Mark Manasse, and Rina Panigrahy. 2008. Design tradeoffs for SSD performance. In *USENIX 2008 Annual Technical Conference (ATC)*, 2008.

[25] Hsin-Yu Chang, Chien-Chung Ho, Yuan-Hao Chang, Yu-Ming Chang, and Tei-Wei Kuo. 2016. How to enable software isolation and boost system performance with sub-block erase over 3D flash memory. *In Proceedings of the Eleventh IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES)*, 2016.

[26] Samsung. Samsung V-NAND Technology. https://www.samsung.com/us/business/oem-solutions/pdfs/V-NAND_technology_WP.pdf. 2018.

[27] Jeong-Uk Kang, Heeseung Jo, Jin-Soo Kim, and Joonwon Lee. 2006. A superblock-based flash translation layer for NAND flash memory. In *Proceedings of the 6th ACM & IEEE International conference on Embedded software (EMSOFT)*, 2006.

[28] Junghee Lee, Youngjae Kim, Galen M. Shipman, Sarp Oral, Feiyi Wang, and Jongman Kim. 2011. A semi-preemptive garbage collector for solid state drives. In *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS*), 2011.

[29] Tae-Sun Chung, Dong-Joo Park, Sangwon Park, Dong-Ho Lee, Sang-Won Lee, and Ha-Joo Song. 2009. A survey of Flash Translation Layer. J. Syst. Archit. 55, 5-6 (May 2009), 332-343.

[30] M.-C. Yang, Y.-M. Chang, C.-W. Tsao, P.-C. Huang, Y.-H. Chang, and T.-W. Kuo. Garbage collection and wear leveling for flash memory: Past and future. In *Proc. SMARTCOMP*, 2014.

[31] Qin Xiong, Fei Wu, Zhonghai Lu, Yue Zhu, You Zhou, Yibing Chu, Changsheng Xie, and Ping Huang. Characterizing 3D Floating Gate NAND Flash: Observations, Analyses, and Implications. ACM Trans. Storage 14, 2, Article 16 (April 2018).

[32] T.-H. E. Yeh et al. Z-interference and Z-disturbance in vertical gate-type 3-D NAND, *IEEE Trans. Electron Devices*, vol. 63, no. 3, pp. 1047-1053, Mar. 2016.

[33] Shuo-Han Chen, Yen-Ting Chen, Hsin-Wen Wei, and Wei-Kuan Shih. 2017. Boosting the Performance of 3D Charge Trap NAND Flash with Asymmetric Feature Process Size Characteristic. In *Proceedings of the 54th Annual Design Automation Conference 2017 (DAC),* 2017.

[34] Arya, P. A Survey of 3D Nand Flash Memory. EECS Int'l Graduate Program, National Chiao Tung University, 2012, pp. 1-11.