IEICE TRANS. INF. & SYST., VOL.E99–D, NO.7 JULY 2016

PAPER

Power Consumption Signature: Characterizing an SSD*

Balgeun YOO[†a)], **Seongjin LEE**[†b)], ***Nonmembers***, ***and*** **Youjip WON**[†c)], ***Member***

SUMMARY SSDs consist of non-mechanical components (host interface, control core, DRAM, flash memory, etc.) whose integrated behavior is not well-known. This makes an SSD seem like a black-box to users. We analyzed power consumption of four SSDs with standard I/O operations. We find the following: (a) the power consumption of SSDs is not significantly lower than that of HDDs, (b) all SSDs we tested had similar power consumption patterns which, we assume, is a result of their internal parallelism. SSDs have a parallel architecture that connects flash memories by channel or by way. This parallel architecture improves performance of SSDs if the information is known to the file system. This paper proposes three SSD characterization algorithms to infer the characteristics of SSD, such as internal parallelism, I/O unit, and page allocation scheme, by measuring its power consumption with various sized workloads. These algorithms are applied to four real SSDs to find: (i) the internal parallelism to decide whether to perform I/Os in a concurrent or an interleaved manner, (ii) the I/O unit size that determines the maximum size that can be assigned to a flash memory, and (iii) a page allocation method to map the logical address of write operations, which are requested from the host to the physical address of flash memory. We developed a data sampling method to provide consistency in collecting power consumption patterns of each SSD. When we applied three algorithms to four real SSDs, we found flash memory configurations, I/O unit sizes, and page allocation schemes. We show that the performance of SSD can be improved by aligning the record size of file system with I/O unit of SSD, which we found by using our algorithm. We found that Q Pro has I/O unit of 32 KB, and by aligning the file system record size to 32 KB, the performance increased by 201% and energy consumption decreased by 85%, which compared to the record size of 4 KB.

key words: *SSD, characterization, power consumption measurement, internal parallelism, I/O unit, page allocation scheme*

1. Introduction

NAND flash memory based SSDs (Solid State Drives) are introduced as a solution to overcome the limitations of HDDs (Hard Disk Drives), such as high power consumption, high noise level, low bandwidth, low IOPS (Input/Output Operations per Second), etc. [1], [2]. Although the bit density of flash memory devices is significantly increased by storing more cells per in^2 and more bits per cell (MLC [3], TLC, and QLC [4]), almost all other aspects of NAND flash memory devices are getting worse: the program time is getting longer, endurance (erase and write cycle) is getting shorter, and retention period (expected time to hold data reliably) is getting shorter [5]. In addition, there are other problems with SSDs, such as slow random write performance and high power consumption [6], [7].

The latency and throughput of a storage device depend heavily on how its internal firmware algorithm handles incoming I/Os, and this boils down to the issue of how to place data blocks on the storage media. In legacy HDDs, the firmware algorithms focus on reducing head movement overhead in reading or writing data; to achieve this, cylinder serpentine [8], surface serpentine [9], and hybrid serpentine [10] methods have been proposed. At the time, one of the key concerns among practitioners was to understand how data blocks are laid out on the storage device. This information can be used to optimize the filesystem layout for a given storage media [11].

Disk layout algorithm is closely link to mechanical characteristics of a given HDD model. Disclosing the disk layout algorithm of HDDs means revealing the manufacture's critical competitive edge. Therefore, the disk layout algorithm is often the most hidden part of the device. Numerous efforts have been dedicated to shed light on understanding the data layout mechanism of storage devices [12]–[14]. For HDDs, these works exploit I/O latency to infer the seek time for a given sequence of I/O operations and subsequently to find out the data layout scheme (e.g., DIG [11]). NAND flash storage device, which is SSDs for desktops and servers or eMMCs for smartphones, consists of multiple NAND flash chips. A write request from the host (e.g., 32 KByte) can be directed to a single chip or can be interleaved across multiple chips in a certain granularity, exploiting a certain type of parallelism. SSDs read data blocks from the flash chip in 4 KByte or larger units. SSD vendors are extremely reluctant to disclose the firmware algorithm of their SSDs, and it is one of the most hidden parts of NAND based storage devices.

The existing works on reverse engineering of HDDs cannot be applied to SSDs because they exploit seek time behavior of the HDDs to understand their internal structure. This work aims at developing a method to derive the internal behavior of SSDs. In particular, we focus on finding the degree of internal parallelism and how it is exploited in laying out data. In this regard, we exploit the power consumption behavior of SSDs. We discover the hidden features of SSDs to improve the performance. As a result, we develop the method, named I/O unit aligning, that improve performance

Manuscript received September 27, 2015.
Manuscript revised January 29, 2016.
Manuscript publicized March 30, 2016.

[†]The authors are with the Division of Computer Science and Engineering, Hanyang University, Seoul 133–791, Korea.

*This paper is an extended version of work published in HotStorage'11, June 14-17, 2011, Portland, USA [15].

a) E-mail: starhunter@hanyang.ac.kr
b) E-mail: insight@hanyang.ac.kr
c) E-mail: yjwon@hanyang.ac.kr (Corresponding author)

DOI: 10.1587/transinf.2015EDP7381

Copyright © 2016 The Institute of Electronics, Information and Communication Engineers

and significantly reduce the energy consumption. In addition, we present the Power Budget as a guideline to evaluate SSDs.

In our previous work [15], we discovered the page allocation scheme of Intel X-25M, the I/O unit of Samsung MXP, and introduced the concept of Power Budget in designing SSDs. In this paper, we delve into the characteristics of two additional SSDs, Samsung 840 and Toshiba Q pro. While the previous work focused on power consumption of write operations and analyzing the result, this work not only considers the read behaviors of SSDs but suggests a formal characterization method to generalize the technique. This paper also introduces a mechanism to improve the performance of SSDs while minimizing the power consumption of the device. This work's contributions are as follows:

- (i) Formal characterization method to find the internal parallelism, I/O unit, and page allocation scheme of SSDs (Sect. 4: SSD Characterization)
- (ii) Analysis of the read behaviors of Intel X-25M and Samsung MXP
- (iii) Analysis of the read and write behaviors of Toshiba Q Pro and Samsung 840
- (iv) Mechansism to improve the I/O performance while minimizing the power consumption via exploiting I/O unit
- (v) Improved Power Budget model.

The remainder of this paper is organized as follows: Sect. 2 describes other works related to this study. The background, such as the SSD architecture, flash memory, channel and way, and FTL (Flash Translation Layer) are briefly discussed in Sect. 3. Section 4 presents three essential SSD characterization algorithms. Experiment environment, validation and case studies on the four SSDs are presented in Sect. 5. Section 6 explains the benefits of aligning the I/O unit of SSDs. Section 7 explains Power Budget. Section 8 concludes the paper.

2. Related Work

There are previous works that exploit seek time of disk arm to determine the sector layout strategy in HDDs [11], [16], [17]. However, the HDDs and SSDs have a different physical structure. HDDs have various mechanical parts (platter, spindle, head, actuator arm, actuator axis, and actuator), while SSDs only have electronical parts. Therefore, these previous studies on HDDs cannot be applied to SSDs.

A number of works characterized the behavior of SSDs through mathematical modeling or via simulation [18]. Yang et al. [19] and Tao et al. [20] used simulation method to determine page size, the address allocation policies, and parallelism. In particular, Tao et al. [20] examined the effect of write buffer size and page size on overall performance of SSDs. Changing the write buffer from 4 MB to 32 MB did not have a significant impact on the throughput (MB/s). However, varying the page size from 1KB to 4 KB resulted in noticeable changes in the average response time (msec) and the throughput (MB/s).

Simona et al. [21] used Flashsim [22] to figure out the discrepancies between the potential performance of SSDs and the observed performance of real-world workloads (Microsoft Research (rsrch0, prxy0, src11, proj2), Harvard University (dea2, akadeasna2, and lair62b), and UMass Trace Repository (fin1, fin2)) [22]. They proposed an SSD performance prediction model. This model shows that SSDs are not utilized to its maximum performance.

Some exploited energy consumption behavior to understand the internal behavior of SSDs. Shin et al. used uFlip [23] to measure the power consumption of SSDs. They connected an electronic resistor between the SSD and power line to collect power consumption through the resistor using Labview; they tried to analyze page allocation scheme of SSDs. SSDs consume high energy in a short time while performing write and erase operations. Although the peak value of power consumption is important in analysis, their method failed to capture them because of the low sampling resolution. Similarly, Seong et al. [24] collected power consumption of real SSDs and HDDs and compared them with power consumption of the in-house developed SSD. With PCI-6259 (the National Instrument's data acquisition board), Seong et al. measured the energy consumption of storage devices in μjoules and used the results as a basis to compare the performance of SSDs. However, they did not provide enough proof that their method can be used to analyze the behavior of SSD. Seo et al. [7] collected power consumption of SSDs using SM2040 (Signametrics PCI digital multimeter), and analyzed power consumption patterns. They measured energy consumption of SSDs while performing random/sequential read/write operations in various sizes. Although their experiment can be used to characterize SSDs, their focus was limited to energy efficiency side of SSDs, according to the workloads. Mohan et al. [25] used data of Grupp et al. [26] to validate their power model. Grupp et al. measured power consumption of SSDs while performing basic operations such as read, program, and erase using high-resolution current probe (Agilent 1147A). Grupp et al. also measured and calculated energy per operation in terms of peak power, average power, and idle power of flash memory. Grupp et al. collected power consumption of SSDs with high sampling rate so as not to lose the peak value. Therefore, unlike Shin et al., they collected all the peak value of power consumption. Also, Grupp et al. suggested Mango FTL to improve responsiveness of SSDs and decrease their power consumption. Park et al. [27] classified the state of SSDs into four categories (active, idle, standby, and sleep) and measured the Ampere (A) for each state using an oscilloscope. However, they presented data only for the in-house developed SSDs which makes it difficult to apply their work to other studies or applications.

A common problem in most of these measurement is a low sampling rate (longer than 1 msec). To accurately capture the energy consumption behavior of SSDs, sampling interval should be set smaller than program time of a NAND

flash, e.g., 900 µs [28]. In this paper, we propose SSD characterization techniques and algorithms to identify internal parallelism, I/O unit, and page allocation scheme of SSDs, with a high sampling rate (less than 2 us). We applied these techniques and algorithms to four real devices and present the results as a case study.

3. Background

3.1 Flash Memory and SSD

The structure of a common flash memory package is shown in Fig. 1 [6]. A flash memory package contains two or more dies (chips), and each die can be selected individually to execute commands independently [29]. Typically, a die is composed of two or more planes and in most flash memories [29], [30], each plane has a page register which is used as a buffer for read and write operations [22]. Page registers support multiple planes that concurrently performs the same operation. This is called plane-level parallelism [31].

In modern SSDs, flash memory packages are organized into multiple groups and each group is allocated a dedicated bus called channel and way. The flash packages that are attached to the different channels can transfer data in parallel manner, and are attached to the ways can transfer data in interleave manner. Flash memory packages are connected in multi-channel and multi-way fashion.

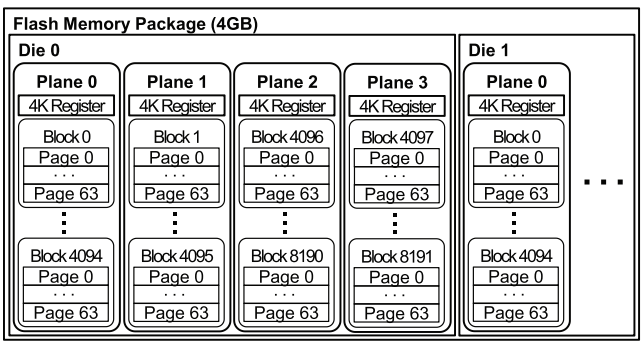


Fig. 1 Internal structure of a Flash Memory Package, based on product specifications on Samsung's K9XXG08UXM series NAND flash

A typical main components of SSDs include host interface, internal buffer (DRAM or SRAM), SSD controller, flash memory controller, and flash memory. Host interface connects the SSD with the host via standard interface (e.g., SATA or IDE). An internal buffer holds data or metadata, which will be written to the flash memory, and stores the address table of flash memory. SSD controller manages the flash memory via the flash memory controller. SSD controller executes firmware, such as FTL, buffer management, ECC, etc. FTL is mainly responsible for three tasks: address mapping [32]–[34], garbage collection [35], [36], and wear-leveling [37], [38]. Flash memory controller manages the flash memory packages in each channel. It has cache register to cache data in channel operations. Flash memory packages in the same channel share cache register. Read and write operations are performed in an interleaved manner [39]. Some SSDs have ECC block for each channel [27].

3.2 Energy Consumption of SSDs

The peak energy consumption of SSDs provides us important information in characterizing the internal structure of SSDs, which is closely related to SSD's degree of parallelism. When data is programmed in parallel to a number of empty flash blocks, the throughput and energy consumption of SSD increases. However, when data is programmed to a empty flash block in serial, the throughput and energy consumption of SSD decreases than parallel case. There is a trade-off between an peak energy consumption of SSDs and their throughput. To characterize the internals of SSDs such as page allocation algorithm, and the number of channels and ways, we exploit the peak energy consumption of device.

Figure 2 illustrates the energy consumption in writing three pages. Writing a page to a NAND flash can be partitioned into three phases: i) sending a command to the command register (C), ii) sending data to data register (D), iii) and programming the NAND page using the content in the data register (P).

There are three layout while programming three pages to the flash memories: All pages are on the same die; spread

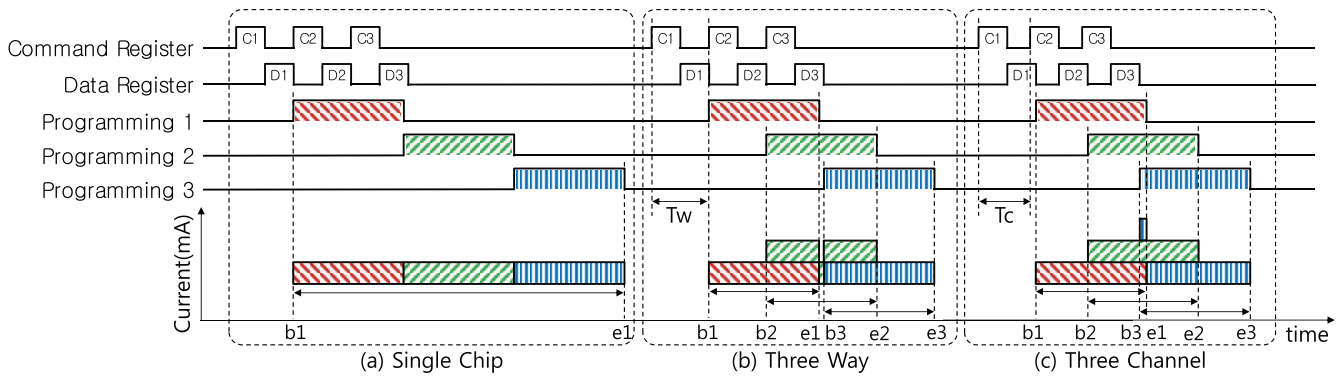


Fig. 2 Page allocation strategy vs. current consumption (b_i: beginning of programming chip i, e_i: end of programming chip i, T_w: way switching time, T_c: channel switching time)

across multiple dies on a channel; and spread across different dies in different channels. Figure 4 (a) shows the first case that three pages are written to the same flash die. Three pages are written sequentially, therefore there can be only one programming activity at a time and this case has longest time. Figure 4 (b) shows the second case where three pages are written to different flash dies attached to the same channel. The process C and D can be serialized, and the flash dies can perform the programming concurrently. We call this way-parallelism. Figure 4 (c) shows the third case that three pages are written to flash dies in different channels. The process C and D can be parallelized, and the latency of this case is the shortest. This is called channel-parallelism.

In one of the SSD models[†], we observed some delays in switching channels [15]. We carefully suspect that this is due to the hardware overhead of channel multiplexing. As shown in Fig. 4 (c), with higher degree of parallelism, the total time for writing three pages decreases. When SSDs use multiple flash memory packages in parallel to increase the performance, the duration of current peak decreases. However, the peak current increases for a brief moment in the number of flash memory packages that use power. Conversely, when a small number of flash memory packages are used to perform an operation, the performance decreases and the duration of current usage increases but the peak current decreases.

The number of packages used in performing read and write operations varies in SSDs. Using high number of packages would result in better performance but at the expense of high power consumption.

4. SSD Characterization

In this paper, we develop characterization method to find three features of SSDs. The first feature is its internal parallelism. SSD has four levels of parallelism (channel, chip, die, and plane). We are especially interested in finding die-level parallelism. Second, we find an read/write I/O unit size of SSDs, which is larger than a page. Third, we will find an page allocation scheme of SSDs. This means finding the location of a flash memory package on which data is written.

While programming write requested data, SSDs show various patterns of peak currents, which are closely related to the number of dies and the I/O unit size. In this paper, we apply our characterization method to four real SSDs to find their three features. We find the number of dies and the I/O unit size by analyzing the energy consumption patterns of SSDs while they perform read and write operations. Those information, combined with the number of flash memory packages and the number of channels, enable us to infer channel/way utilization and page allocation scheme of SSDs.

4.1 Internal Parallelism and I/O Unit

In analyzing energy consumption of SSDs, we focus on how its peak current changes when write requested data size increases. In a flash memory, a page is the smallest unit size in reading and writing data. Page sizes are different (4 KB, 8 KB, or 16 KB), depending on flash memory packages. If a flash memory package has multi-flash memories (chip or die), each die can operate page programming concurrently. Therefore, the number of dies equal to the number of concurrent programming operations. A large write request are divided into *page size* × (*no. of dies*) to be allocated to each package. SSDs that are performing program operation has higher peak current (active state peak current) than SSDs in idle state. SSDs show the minimum active state peak current (m_{peak}) when it is performing page-size write, and shows the maximum active state peak current (M_{peak}) when all packages are activated. When SSDs are using one package with multiple dies to program *page size* × (*no. of dies*), its peak current is same as when they are performing page-size write because program operation perform programming in an interleaved manner. Therefore, if write requested data size is increased, peak current of SSDs increase in steps from m_{peak} to M_{peak}.

In some SSD controllers read and write in units that are larger than a page. They read or write multiple pages, simultaneously, as a single operation to reduce the number of read/write operations. We define this unit as the I/O unit of SSDs. As explained above, the peak current of SSDs change in steps. Generally, as the number of dies in a flash memory package increases, the tread depth of each step increases, and as the size of I/O unit increases, the number of steps decreases. Figure 3 (a) and Fig. 3 (b) show how the tread depth of steps change when the number of dies changes in an SSD which has 4 packages and 1 page I/O unit. Figure 3 (a) shows peak current of a flash memory package with 1 die. If the write size is 1 page, the peak current is m_{peak} because I/O unit is 1 page. If the write size is 4 pages, the peak current is M_{peak}. Therefore, the number of steps is 4, and the tread depth of a step is 1 page. Figure 3 (b) shows peak current of a flash memory package that has 2 dies. In this case, the peak current is m_{peak} for both 1-page and 2-pages writes because two dies are activated concurrently. If the write size

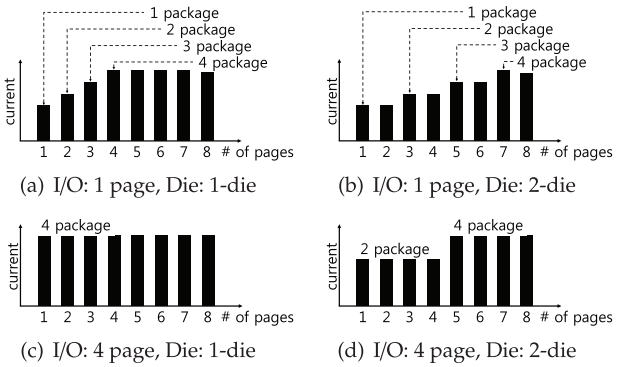


Fig. 3 Changes in peak current according to I/O unit sizes and the number of dies in an SSD with four packages (I/O: I/O Unit, Die: number of Dies per package, Package: number of active packages)

[†]Samsung SSD 840 PRO

is larger than 6 pages, the peak current is M_{peak}. Therefore, the number of steps is 4, and the tread depth of a step is 2 pages. Figure 3 (c) shows how the number of steps changes when I/O unit changes from 1 page (Fig. 3 (a)) to 4 pages in an SSD with 1 die. Even when the write size is 1 page, the SSD has M_{peak} because I/O unit is 4 pages. When write size larger than 4 pages and smaller than 9 pages, two write operations are performed serially with no peak current change. Therefore, there are no steps in this case. Figure 3 (d) shows how the number of steps changes when I/O unit changes from 1 page (Fig. 3 (b) to 4 pages in an SSD with 2 dies. The peak current is $m_{peak} \times 2$ for both 1-page and 4-pages writes. When the write size is larger than 5 pages, the peak current increases to M_{peak}. Therefore, the number of steps is 2, and the tread depth of a step is 4 pages.

We can infer the number of dies in a flash memory package (N_{die}) from the tread depth of a step (D_{step}) and the number of steps (N_{step}) both obtained from our experiment, and the number of flash memory packages ($N_{package}$) that can be obtained from the vendor specifications. By multiplying N_{step} and D_{step}, we can get the number of pages that an SSD can program simultaneously. By dividing this value by $N_{package}$, we can get the number of pages which can be programmed simultaneously to a package. This value is equal to N_{die} (Eq. (1)).

$$N_{die} = N_{step} \times D_{step}/N_{package} \quad (1)$$

$$I/O\ Unit = N_{package}/N_{step} \times N_{die} \quad (2)$$

Since, the size of I/O unit is fixed by firmware of SSDs, inferring the size of I/O unit is essential to performance optimization. We can infer the I/O unit information from N_{step}, $N_{package}$, and N_{die}. The number of flash memory packages that belongs to one step can be obtained by dividing $N_{package}$ by N_{step}. The number of dies that belongs to one step can be obtained by multiplying this value by N_{die} (Eq. (2)). If we substitute N_{die} in Eq. (2) with Eq. (1), we can see that the I/O unit is identical to D_{step}.

Through our experiment, we can obtain N_{step} and D_{step}. We used sequential write workload and started write size at the page size of an SSD. Write size is increased in multiple of page size until the peak current of the SSD reached its M_{peak}, which was obtained before the experiment by performing 1 MByte sequential write; 1 MByte sequential write is sufficient size to map the I/O to all flash memory packages in all target SSDs. We then measured the peak current value for each write size. The experiment terminated when the peak current reached M_{peak}. N_{step} and D_{step} can be found by plotting the results on a graph.

We already defined the I/O unit as D_{step}. However, this definition only applies to write I/O units and a separate experiment is needed to determine read I/O unit size. When a read request is issued by the host, the SSD controller looks for the physical location of the requested data. After locating the requested data, the flash memory controller reads I/O unit data from the flash memory to the cache register. After reading I/O unit data to the cache register, the SSD controller reduces power supplied to the flash memory from active level to standby level because the flash memory is not being used while the data is transferred from the cache register to I/O buffer and from I/O buffer to the host. Because the power stage changes from active to idle, the waveform of the current falls. Therefore, it is possible to find the read I/O unit size by the number of peak currents in the waveform.

Read and write I/O unit sizes refer to the actual read and write units sent to the flash memory by the SSD controller. Information on I/O unit size is essential to improving the performance of SSDs. If the OS knows the I/O unit of the SSDs, interface bottleneck can be reduced by adjusting the basic I/O unit of the OS to the I/O unit of SSDs. Also, by changing a random I/O to a sequential I/O, fragmentation of SSDs can be reduced.

4.2 Page Allocation Scheme

In this section, we present characterization method to find page allocation scheme of SSDs. In order to do this characterization, we need information from vendor specifications, such as number of channels ($N_{channel}$) and number of packages ($N_{package}$). We also need experiment results, such as D_{step}, N_{die}, and peak current duration. Our method consists of two steps. First, we find the physical structure of an SSD. For example, if an SSD has 8 channels and 16 packages, then 2 packages are connected in each channel and the number of dies in each package is N_{die}. Second, we infer I/O allocation scheme. SSD controller chooses target packages to allocate write requested data. If write requested data are allocated to multiple packages, the target packages are divided into two cases:

Case 1: Target packages share a channel: If write requested data are allocated to a package, data is transferred to the register of package through the channel, and this process occupies the channel until the transfer is complete. Therefore, when target packages share a channel, delay occurs during register transfer time. From this result, we can conclude that if one channel is used to allocate write requested data, then the performance of SSD is reduced by register transfer delay, but since packages perform programming in an interleaved manner, peaks in current are also reduced. In such case, we use case 2.

Case 2: Target packages do not share a channel: In this case, after first target package is selected, the next I/O is allocated to another channel. This incurs only channel switching delay which is less than the register transfer delay [15]. From this result, we can conclude that when multiple channels are used to allocate I/Os, the performance of SSD increases; since multiple packages operate concurrently, this also increases peaks in current consumption than case 1.

We can calculate an I/O duration of an SSD using register delay or channel switching delay. By comparing the calculated values to the actual I/O duration obtained from an experiment, we can infer the SSD's page allocation scheme. We have to consider the three cases that write requested data are allocated to multiple packages. We explain the three cases by using the example in Fig. 4 which has 4 packages.

Table 1 Characteristics of SSDs

Model Name	Interface	Capacity	Type	R/W Performance	Channel	DRAM Size	Release Date
Intel X-25M	SATA 2	80 GB	MLC	250/70 MB/s	10	16 MB	2008.09
Samsung MXP	SATA 2	128 GB	MLC	220/200 MB/s	8	128 MB	2009.04
Toshiba Q Pro	SATA 3	128 GB	MLC	554/512 MB/s	4	-	2013.03
Samsung 840	SATA 3	250 GB	TLC	530/250 MB/s	8	512 MB	2012.10

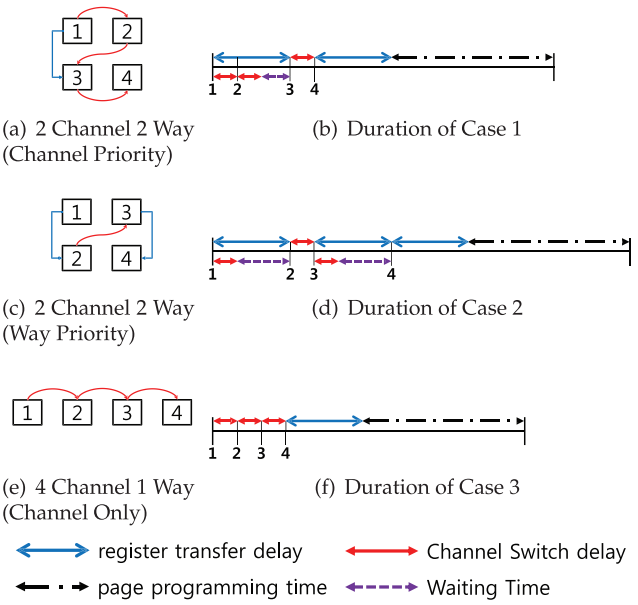


Fig. 4 Configuration cases of page allocation scheme

Figure 4 (a) shows the first case that four packages connected with 2-channel 2-way configuration. When data is received, the package #0 of first channel is selected and the first channel is preempted which incurs register transfer delay. When the package #0 of second channel is selected and the second channel is preempted, then there is channel switch delay and register transfer delay. The package #1 of first channel cannot be selected immediately because the package #0 has preempted the first channel. The package #1 of first channel must wait before it can performs register transfer operation. When the package #1 of second channel is selected, channel switching delay occurs but not register transfer delay because the operation on the package #0 of second channel is complete. Therefore, the package #1 of second channel can perform register transfer without waiting. We call this channel priority allocation method.

Figure 4 (c) shows the second case that 4 packages connected with 2-channel 2-way. The package #0 of first channel preempts the channel which incurs register transfer delay. The package #1 of first channel cannot be selected, because the package #0 of first channel preempted the channel. The package #1 of first channel must wait before it performs register transfer. Channel switching delay occurs to select the package #0 of second channel and register transfer delay also occurs when the package preempts the channel. The package #1 of second channel cannot be selected because the package #0 of second channel preempted the channel. The package #1 of second channel must wait before it performs register transfer operation. This case has longest duration of three cases, as shown in Fig. 4 (d). We call this way priority allocation method.

Table 2 Internal parallelism and I/O unit

Name	N_{step}	W_{step}	N_{die}	I/O Unit	M_{peak}
X-25M	20	1 page	1	-	530 mA
MXP	4	8 pages	2	32 KB	400 mA
Q Pro	4	4 pages	4	32 KB	644 mA
840	1	1 page	1	-	650 mA

Figure 4 (e) shows the third case that 4 packages connected with 4-channel 1-way. In this case, there is no waiting time, unlike Fig. 4 (a) and Fig. 4 (c). There is only channel switching delay. Whenever there is waiting, it increases the duration of I/O which decreases the performance of SSD. Also, since power is continuously supplied to the device even during the waiting period, it leads to power inefficiency. This case, which does not have waiting time, has shortest duration of three cases as shown in Fig. 3 (f). We call this channel only allocation method.

We can infer the page allocation scheme of SSDs as following steps. First, calculate the duration of three cases. Second, compare the calculated duration with the actual duration obtained from an experiment. The case which has the closest duration of the experiment is the page allocation scheme of the target SSD. Third, expand the page allocation scheme to other write request size. Detailed analysis with four real SSDs are addressed in Sect. 5

5. Case Study

In this paper, four SSDs are studied: Intel X-25M, Samsung MXP, Toshiba Q Pro, and Samsung 840. Intel X-25M has 10 channels; Samsung MXP has 8; Toshiba Q Pro has 4; and Samsung 840 has 8 channels. The size of I/O buffer varies as shown in Table 1. Toshiba Q Pro does not have DRAM I/O buffer on PCB board. We conjecture that Toshiba Q Pro has internal DRAM in controller chip. In this section, various features of SSDs, such as power consumption in idle state, internal parallelism, I/O unit, and page allocation scheme, are described.

Table 2 summarizes the results of our experiment. Workloads for each SSD started at 4 KB and increased by 4 KB to 160 KB with X-25M and to 128 KB with MXP and Q Pro. As we increased the write size, the peak current increased. The intervals at which the peak current increments are varied by the SSDs. With X-25M, the peak current increased every time the size increased by 4 KB. MXP and Q Pro increases its peak current on every 32 KB increase in the write size.

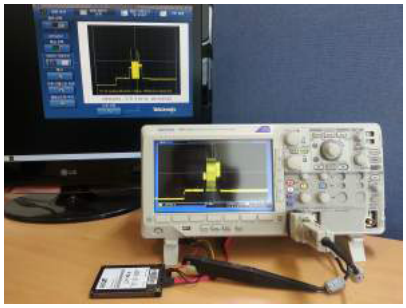


Fig. 5 Measurement environment

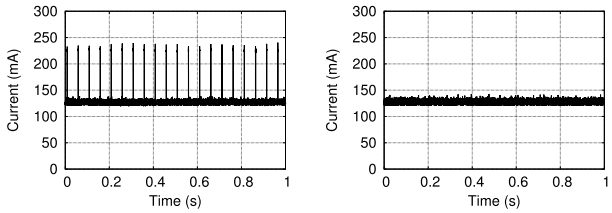


Fig. 6 Intel X-25M standby mode

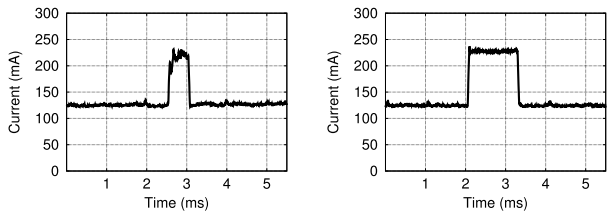


Fig. 7 Intel X-25M Write Caching On/Off

Table 3 SSD initialization commands

Name	Option	Flag
Read-ahead	-a	0 to 248
Look-ahead	-A	0/1
Drive write-caching	-W	0/1
Drive standby	-y	-

5.1 Experiment Environment

5.1.1 Measurement Hardware

Figure 5 shows the experiment environment which consists of host system, an oscilloscope, and the target SSD. A high-resolution current probe is used in measuring current of SSDs between the target and the host system. Tektronix DPO3012 oscilloscope is used in data collection. In general, a flash memory's program time is shorter than 900 µs [28]. In this paper, sampling rate of the oscilloscope is set at 2 µs in write operations and 4 µs in read operations to measure current without losing the peak value. The host system, which creates workload to the target SSD, has Intel Dual Core2 2.9 GHz CPU and 4 GB main memory. The host system was loaded with Linux 2.6.39 and we opened SSDs as a raw device to reduce the noise caused by the file system. We calculate the power (watt) by multiplying current with input voltage which is 5 V, and the energy (joule) is calculated by multiplying the 'averaged' power with the time.

5.1.2 Initialization SSD

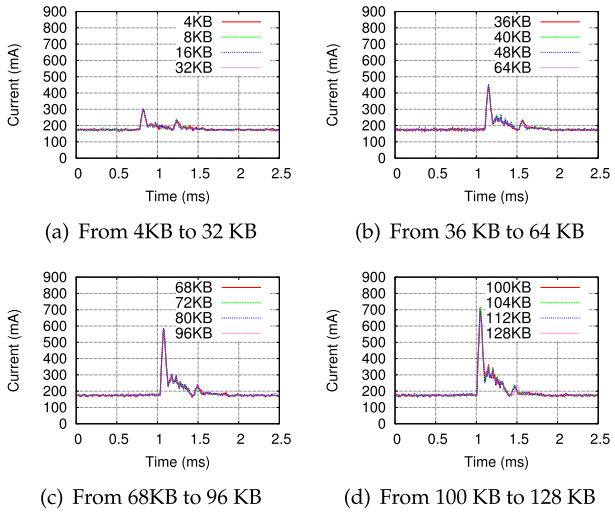
As SSDs are used extensively, the number of invalid pages increases and SSDs become "dirty". When "dirty" SSDs perform a workload, garbage collection will be performed in the background. To prevent this background operation, which might interfere with the test, "dirty" SSDs should be initialized before running an experiment. In this paper, secure erase technique [40] (ANSI ATA and SCSI disk interface specific disk purging commands that are performed internal of the disk) is used to initialize SSDs.

In this paper, initialization refers to setting the target SSD to the right state before performing experiments. When SSDs are used without initialization, there is too much noise which makes difficult to separate small sized I/Os from the noise. Some features of SSDs that can interfere with performing workload, such as read-ahead, look-ahead, and write buffer, and these are turned off before the experiment.

Figure 6 (a) shows power consumption of Intel X-25M when it is in idle state. It shows 240 mA peaks at 50 msec intervals. Without removing these peaks, it is difficult to analyze the results because these peaks in idle state are similar to the peak values of performing a 12 KB write operations. Figure 6 (b) shows power consumption of Intel X-25M after a standby command was sent with hdparm [41] using -y option. It shows that 240 mA peaks are removed. In some SSDs, write request data is not written to the flash memory but is recorded only on an I/O buffer (DRAM or SRAM). In this case, current value obtained in the experiment is not from the flash memory. Figure 7 shows the two energy consumption patterns of X-25M, Fig. 7 (a) shows the power consumption of the SSD when the write requests are written to the I/O buffer. Figure 7 (b) is when the requests are written to the flash memory. To prevent write request data from being recorded on an I/O buffer, we disabled the device's write buffer by a command (hdparm) before the experiment. The initialization commands are summarized in Table 3.

5.1.3 Data Sampling

In this paper, we collect electric current, which is the amount of charge flowing through the conductor per unit time. Magnetic or heat flow can interfere with measuring current and noise was initially found in the oscilloscope. For efficient analysis, we used moving average on the collected data. Very high and short-lasting peaks observed on raw data are lost after use moving average; this is not a problem because what we want to observe is the gap between the peaks and the pattern of power consumptions, not the original peak



(a) From 4KB to 32 KB (b) From 36 KB to 64 KB

(c) From 68KB to 96 KB (d) From 100 KB to 128 KB

Fig. 8 Power consumption of Toshiba Q pro for write operations with increasing request size

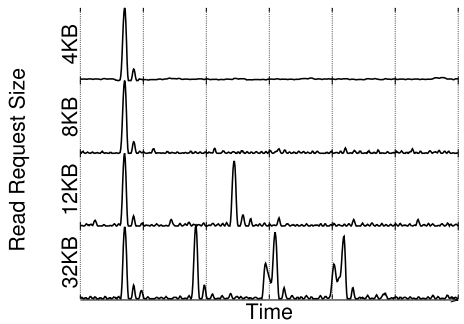


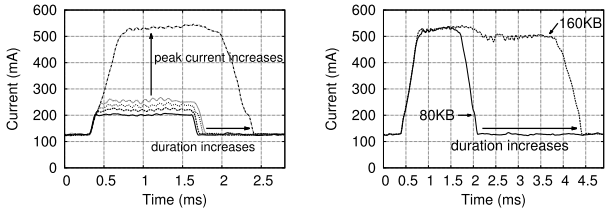
Fig. 9 Power consumption of Toshiba Q pro for read operations

values. In this paper, two window sizes were used in calculating moving averages. First, window size of 20 was used to find the internal parallelism of SSD, where the exact peak values are important. Second, window size of 100 was used to find the I/O unit, where patterns of results are more important than the exact peak values.

5.2 TOSHIBA Q Pro

Figure 8 shows power consumption of Toshiba Q Pro performing write operations in sizes from 4 KB to 128 KB in increment of 4 KB. Since Toshiba does not disclose page size of Q Pro, we used the method presented in Chen's paper [31] to find the information. From this method, we found page size of Q Pro to be 8 KB. Therefore, D_{step} of Q Pro is 4 pages (32 KB). From Eq. (1) and Eq. (2), we can obtain the following parameters: N_{die} is 4, and write I/O unit is 32 KB.

Figure 9 shows power consumption of Q Pro performing read operations in various sizes, from 4 KB to 32 KB, in increments of 4 KB. The result shows that as the size of read operation increases in increment of 8 KB, power consumption pattern of 8 KB read is repeated: twice with 12 KB, three times with 20 KB, and 4 times with 32 KB read. From this result, we conclude that read I/O unit of Q Pro is 8 KB



(a) From 4KB to 80 KB (b) From 80 KB to 160 KB

Fig. 10 Power consumption of Intel X-25M for write operations with increasing request size

which is different from its write I/O unit.

I/O unit and page table has inverse proportional relationship; thus, larger the I/O unit, smaller the size of page table. Some benefits of keeping I/O units large are that merging pages can be simplified and overhead of allocation a free page can be reduced. However, when there are intermittent write requests from the host, free page pool is soon depleted and the overhead in write operation increases because writes are done in large I/O unit. If FTL takes advantage of write cache in the device, write requests can be merged to form a large chunk of data with the size of I/O unit. In the case of read requests, the device does not have such opportunity to merge the I/O requests in the cache. Thus, at each read request, one has to read in the size of I/O units. As a result, Toshiba Q Pro has different size of read and write I/O unit to improve the I/O performance.

Toshiba Q Pro has 4 channels and 4 flash memory packages. N_{die} is 4. We can infer its configuration as follows: Q Pro has 4 channels; each channel is connected to one flash memory package with way; and each package has 4 dies. Write I/O unit of Q Pro is 4 pages (32 KB) which is allocated to one package (4 dies). 8-page write request is allocated to 2 packages; 12-page write request is allocated to 3 packages; and 16-page write request is allocated to 4 packages. Therefore, the maximum size that can be programmed concurrently is 128 KB ($4 \times 4 \times 8\ KB$) in Q Pro. Since up to four pages are allocated to a single package (i.e., channel), we can conclude that Q Pro uses *way priority* page allocation scheme.

5.3 Intel X-25M

Figure 10 (a) shows power consumption of Intel X-25M performing write operations in sizes, from 4 KB to 80 KB, in increments of 4 KB. From Eq. (1) and Eq. (2) of Sect. 4.1, we can obtain the following parameters: N_{die} is 1, and write I/O unit is 4 KB. Current duration of X-25M increased on average, 33 µs per 4 KB increase in write size, until the write size reaches to 80 KB, except when the write size increased from 80 KB to 84 KB; in this case, the duration increased 1.4 ms as shown in Fig. 10 (b).

Figure 11 shows power consumption of X-25M performing read operations in various sizes, from 4 KB to 16 KB, in increments of 4 KB. The result shows that as the size of read operation increases in increments of 4 KB,

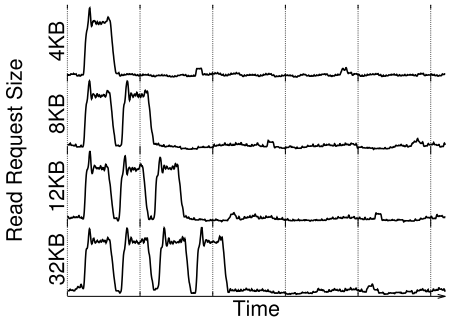


Fig. 11 Power consumption of Intel X-25M for read operations

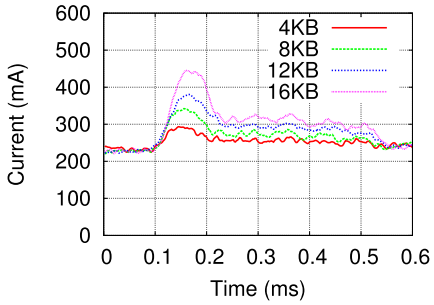
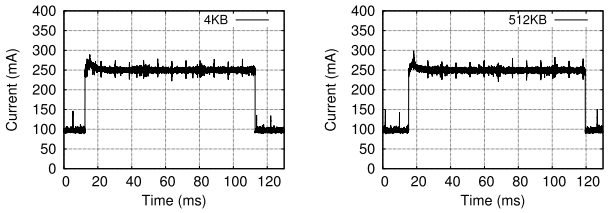


Fig. 12 Power consumption of Samsung 840 for write operations with increasing request size



(a) SAMSUNG 840 4KB Read (b) SAMSUNG 840 512KB Read

Fig. 13 Power consumption of Samsung 840 for read operations with 4 KB and 512 KB

power consumption pattern of 4 KB read is repeated: twice with 8 KB, three times with 12 KB, and 4 times with 16 KB read. From this result, we can conclude that read I/O unit of X-25M is 4 KB which is same as its write I/O unit. With small I/O unit, each channel processes small I/Os, resulting in high performance. A disadvantage is its high power consumption due to a higher number of channels in use.

Intel X-25M has 10 channels and 20 flash memory packages, and N_{die} is 1. From this, we can infer X-25M's flash memory configuration as follows: X-25M has 10 channels; each channel is connected to 2 flash memory packages with way; and each package has one die. The maximum size that can be programmed concurrently is 80 KB ($20 \times 1 \times 4\ KB$) because X-25M allocates page (4 KB) to each die.

X-25M has two types of delay. In our experiment, when we increased write size by 4 KB, duration of peak current increased 33 µs on average. When we increased write size from 80 KB to 84 KB, duration of peak current increased by 1.4 ms. From this result, we can infer that X-25M allocates 84 KB in the following 3 steps.

The first 40 KB is allocated to package #0 in each channels, 4 KB per channel. Each package preempts the channel which incurs register transfer delay. This channel preemption time is 82 µs [30], because this delay is due to 4 KB transfer.

The second 40 KB is allocated to the package #1 in each channels, 4 KB per channel. To select package #1 in each channel as target, there needs to be 10 channel switching. Since channel switching delay in X-25M is 33 µs, 10 channel switching delays ($33\ \mu s \times 10$) is longer than the channel preemption time in the first step (82 µs). Therefore, the second 40 KB is allocated without waiting.

The last 4 KB is allocated to the package #0 of channel #0. However, since the target package is busy with 82 µs register transfer and 900 µs page program, the last 4 KB must wait until the prior operations are finished. The waiting time of the last 4 KB is 322 µs ($(900\ \mu s + 82\ \mu s) - (33\ \mu s \times 20)$). Experiment result took about 1ms longer than our calculation. We can try to guess that page program performance is slower than the vendor specification or there is unknown delay between the continuous program operations. We can conclude that the X-25M uses *channel priority* page allocation scheme.

5.4 SAMSUNG 840

SAMSUNG 840 has 8 channels and 8 flash memory packages, and N_{die} is 1. 840 has very simple page allocation scheme similar to the X-25M. The write data is allocated on a page in each channel. In addition, the maximum size of the concurrent writes are 32 KB. Figure 12 shows power consumption of Samsung 840 performing write operations in sizes, from 4 KB to 16 KB, in increments of 4 KB. The peak current increased by 50 mA per 4 KB increase in write sizes. Its maximum peak current was 650 mA, which was reached while performing 32 KB write. The peak current values increased 8 times while performing the above mentioned write operations. From the result, we can obtain the following parameters: N_{step} is 8, and D_{step} is 1 page (4 KB). From Eq. (1) and Eq. (2) of Sect. 4.1, we can obtain the following parameters: N_{die} is 1, and write I/O unit is 4 KB.

If only the write results are considered, 840 has similar result with X-25M. Figure 13 shows power consumption of 840 performing read operations with 4 KB and 512KB. The result shows that as the size of read operation increases, power consumption pattern is not changed. Furthermore, energy consumption of read operation is too much high. For example, 4 KB energy consumption of read operation with Q Pro is 1.97 mJ, however, 840 is 6,325 J.

From the result, we can conclude that the read I/O unit size of Samsung 840 is something larger than the I/O unit of write operation. The specification of 840 tells us that the device uses TLC NAND Flash memory and has read and write performance of 530 MB/sec and 240 MB/sec, respectively. On the other hand, 840 pro which is based on

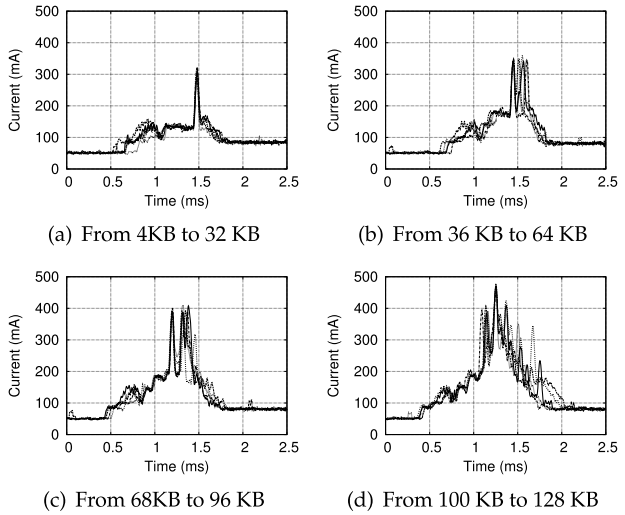


Fig. 14 Power consumption of Samsung MXP for write operations with increasing request size

840 uses MLC NAND and has read and write performance of 540 MB/sec and 520 MB/sec, respectively. One of the main reason for the slower write performance in 840 is that it is using TLC NAND Flash memory while having similar read performance. Since TLC devices are saving three bits in a cell, it has to be more careful in incrementing the voltage to set the bits. One way to deal with it is to introduce more steps used in Incremental Step Pulse programming (ISPP) [42].

5.5 SAMSUNG MXP

Figure 14 shows power consumption of Samsung MXP performing write operations in sizes, from 4 KB to 128 KB, in increments of 4 KB. Using Eq. (1) and Eq. (2) of Sect. 4.1, we can obtain the following parameters: N_{die} is 2, and write I/O unit is 32 KB. We observed an unusual trait in MXP's power consumption. As shown in Fig. 14, the peak current of MXP does not return to idle state power consumption after completing the write operations but is maintained at 80 mA for a while. The full graphs are omitted to save space but the tail of these 80 mA state lasts about 5 seconds. Another noticeable point of MXP is its very low power consumption level during idle state compared to Q Pro, X-25M, and 840. From these two points, the power consumption technique of MXP can be inferred as follow: MXP reduces idle state power consumption by completely blocking power supply to the flash memory when it is not handling an I/O. This may result in standby delay in MXP unable to respond immediately to incoming I/Os resulting in low throughput. In order to avoid low throughput, MXP maintains standby power condition of 80 mA to be ready for any additional I/Os after completing an I/O, so that it can perform I/Os continuously without standby delay. We conclude that MXP reduces power consumption to the extreme in an idle state, when it is not handling any I/O, but once an operation is completed, it consumes more power than needed to be ready for future I/Os.

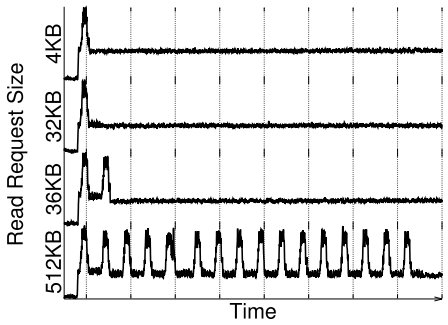


Fig. 15 Power consumption of Samsung MXP for read operations

Figure 15 shows power consumption of Samsung MXP while performing read operations of 4 KB, 32 KB, 36 KB, and 512 KB in size. The result shows that waveform for 4 KB and 32 KB are the same. For 36 KB, waveform for 32 KB is repeated twice; For 512 KB, waveform for 32 KB is repeated 16 times. From this result, we can infer that read I/O unit of MXP is 32 KB which is same as its write I/O unit.

Samsung MXP has 8 channels and 16 flash memory packages, and N_{die} is 2. Configuration of MXP can be estimated as follows: MXP has 8 channels; each channel is connected to 2 flash memory packages with way; and each package has two dies. The maximum size that can be programmed concurrently is 128 KB ($16 \times 2 \times 4\ KB$) because MXP allocates one page (4 KB) to each die.

Write I/O unit of MXP is 8 pages (32 KB), which is allocated to 4 packages (8 dies). There are three ways to allocate 32 KB, depending on the channel and way usage: (i) 2-channel 2-way, channel priority allocation method, (ii) 2-channel 2-way, way priority allocation method, (iii) 4-channel 1-way, channel only allocation method.

When we exploited write I/O unit size, the duration of peak current was measured at 1.12 ms. Considering that other NAND flash memories released about the same time as MXP take about 900 µs [30] to program 4 KB, write speed of MXP is too fast. Therefore, we assume that MXP allocates 4 KB to 2 dies in each package and programs 8 KB concurrently by using the internal command [29]. Based on this assumption, we used 164 µs (82 $\mu s \times 2$) for register delay, which is the time it takes to transfer 8 KB, and used 900 µs for page programming time, which is the time it takes to program 4 KB.

We consider three cases to find the MXP' page allocation scheme; 2-channel 2-way with channel priority, 2-channel 2-way with way priority, and 4-channel 1-way with channel priority. In first case (2-channel 2-way with channel priority allocation method), when data is received, the package #0 of first channel is selected and the first channel is preempted which incurs register transfer delay. When the package #0 of second channel is selected and the second channel is preempted, channel switching delay and register transfer delay occur. The package #1 of first channel

cannot be selected right away because the package #0 has preempted the first channel. The package #1 of first channel must wait 98 µs (164 μs − (33 $\mu s \times 2$)) before it performs register transfer operation. When the package #1 of second channel is selected, channel switching delay occurs but not register transfer delay because channel preemption by the package #0 of second channel has already finished (98 μs + (33 $\mu s \times 2$)). Therefore, the package #1 of second channel can perform register transfer without waiting. This case takes 1.261 µs in total.

In second case (2-channel 2-way with way priority allocation method), the package #0 of first channel preempts the channel which incurs register transfer delay. The package #1 of first channel cannot be selected because the package #0 of first channel preempted the channel. The package #1 of first channel must wait 131 µs (164 µs − 33 µs) before it performs register transfer. Channel switching delay occurs to select the package #0 of second channel and register transfer delay also occurs when the package preempts the channel. The package #1 of second channel cannot be selected because the package #0 of second channel preempted the channel. The package #1 of second channel must wait 131 µs (164 µs − 33 µs) before it performs register transfer operation. This case takes 1.425 µs in total.

In third case (4-channel 1-way with channel only allocation method), there is no waiting time, unlike first and second cases. There is only channel switching delay. Whenever there is waiting, it increases the duration of I/O which decreases the performance of SSD. Also, since power is continuously supplied to the device even during the waiting time, it leads to power inefficiency. This case, which does not have waiting time, takes 1,163 µs. This value is closest to experiment result, 1.120 ms. Therefore, we can conclude that MXP uses 4-channel 1-way to allocate I/O unit sized data.

Another unique feature in MXP is its Read-Modify-Write operation. Figure 16 shows power consumption of Samsung MXP while performing 4 KB write operation and 32 KB read operation. The waveforms for the two operations are almost identical for the first 0.2 ms. The duration of this identical waveform decreases as the write size increases, until the write size reaches 28 KB. For 32 KB write, there is no identical waveform with the read operation at the beginning. However, 36 KB write operation shows similar beginning waveform with 4 KB write. From this result, we can estimate that MXP always writes in 32 KB units to flash memory. If write requested data size is smaller than 32 KB, then MXP reads data from write address to make up 32 KB, places the I/O in I/O buffer, and processes it.

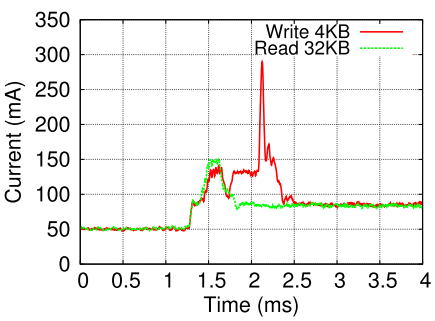


Fig. 16 Proof of read-modify-write

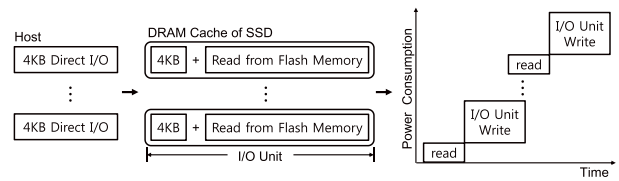
6. Breakdown of SSD Energy Consumption

We use data from various cases to make a formal characterization method. Experimental results are obvious; however, our characterization method needs verification to be reliable. The I/O unit which is discovered with our characterization method is an important factor that could affect the performance and energy consumption of SSDs. We have to verify the I/O unit by appropriate experiments using the I/O unit. We expect that SSDs can improve the performance and reduce the energy consumption.

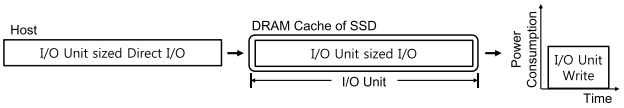
We can consider two cases when performing a large sized write operation in SSDs which has I/O unit: first, performing direct I/O with page size; and second, performing direct I/O with I/O unit size. Figure 17 (a) shows the first case. SSDs extend the write data from page size to I/O unit size in the write cache. The additional data is filled with read data from the flash memory. Therefore, SSDs write the I/O unit sized data to the flash memory whenever a page size write is performed.

Figure 17 (a) shows the second case. SSDs write the I/O unit sized data to the write cache. There are no additional read operation and data extension. Therefore, SSDs will be only consuming the energy of the I/O unit sized write operation. This case has less energy consumption than first case. As a result, if aligning the record size with I/O unit of SSDs in write operation with huge file, we can maximize the performance and minimize the energy consumption.

We conduct validation experiment for I/O unit as follows: The record size is increased from 4 KB to 64 KB and direct I/O is used. We use IOzone as workload generator. And, we acquire performance and energy consumption. The experiment targets used are two I/O unit SSDs (Q Pro, MXP) and two non I/O unit SSDs (840, X-25M). Our workload is as follows: File Size 128 MB, record size is in-

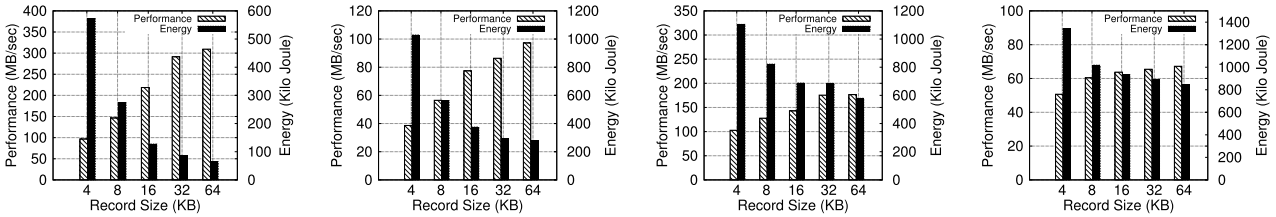


(a) Power Consumption of 4KB sized writes



(b) Power Consumption of I/O Unit sized write

Fig. 17 Power consumption of I/O unit sized write and non I/O unit sized write



(a) Toshiba Q Pro: I/O Unit (b) Samsung MXP: I/O Unit (c) Samsung 840: Non I/O Unit (d) Intel X-25M: Non I/O Unit

Fig. 18 Performance and energy consumption of I/O unit and non I/O unit SSDs

Table 4 Validation result of I/O unit

SSD		Increment/Decrement Ratio	
		Performance (MB/sec)	Energy Consump. (KJ)
I/O Unit	Q Pro	201%	85%
	MXP	123%	71%
Non I/O Unit	840	71%	37%
	X-25M	29%	33%

creased from 4 KB to 64 KB in multiples of two, direct I/O, and sequential write. We expected that the performance is greatly improved and the energy consumption decrease significantly with the increase of the record size in the I/O unit SSDs (Q Pro, MXP).

Figure 18 shows performance (MB/sec) and energy consumption (Kilo Joule) of four SSDs. Table 4 shows ratio of performance and energy consumption when the record size increased from 4KB to I/O unit size. Figure 18 (a) and Fig. 18 (b) show the result of I/O unit cases, and Fig. 18 (c) and Fig. 18 (d) show the result of non I/O unit cases. The I/O unit size of Q Pro and MXP are same as 32 KB.

The result shows that I/O performance of the one with I/O unit is increased to 162% on the average, and the I/O performance of the non I/O unit SSDs increased to 50% on average. In terms of energy consumption, the energy consumption of the I/O unit SSDs reduced to 78% on average, and the energy consumption of the non I/O unit SSDs reduced to only 35% on average. From the result it can be said that Toshiba Q Pro has the I/O unit.

In general purpose systems, aligning I/O requests to the I/O unit size of SSD is not possible because internal configurations of SSDs are proprietaries of manufacturers and they are not willing to disclose the information. The proposed method in this paper provides a technique to expose the internal configuration of SSD that is I/O unit size, which can be exploited in RAID storage system. Typically, the *stripe size* of RAID is carefully determined after thorough analysis of given workload and I/O characteristics. Although the process of optimizing the RAID is tedious, but the fact that it is a one-time effort relieves management and deployment overhead. Our method of finding the I/O unit size also needs to be performed only once. As shown in Fig. 18, the use of the proposed method not only allows increasing the performance by aligning the *stripe size* to I/O unit size of SSD but also decreases the overall operation cost of storage systems.

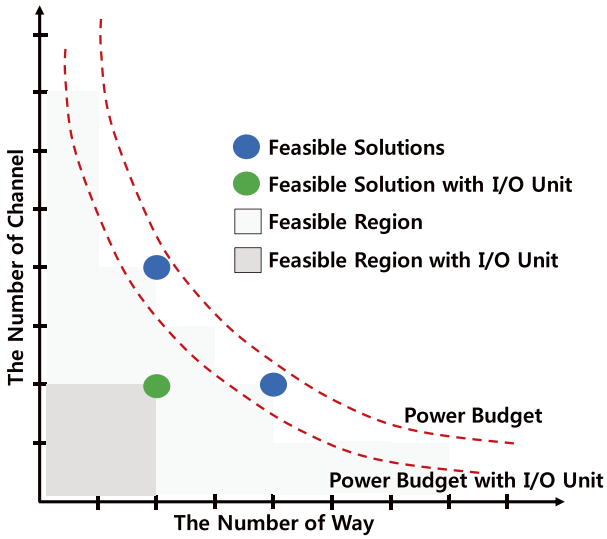


Fig. 19 Power budget and feasible page allocations

7. Power Budget

In previous work, we warned excessive use of channel and way of SSDs [15]. As we can see in related works, the peak current is increased excessively when the program operation is performed concurrently in too many NAND chip. The excessive peak current can cause supply voltage drop, ground bounce, signal noise, black-out, and etc, which can lead to unreliable SSD operation [43]. Therefore, we propose a metric called Power Budget, which specifies the maximum tolerable peak current for SSDs' operations.

The previous version of Power Budget only served the purpose to prevent the excessive simultaneous parallel use of resources. However, it is possible to have better performance and lower energy consumption while using less parallel resources by exploiting the I/O unit aligning in Sect. 6. Therefore, it is able to apply more strict criteria in the Power Budget. Figure 19 shows the new Power Budget. The x-axis is the number of way, and the y-axis is the number of channel. The new Power Budget proposes not only the use of balanced parallelism level also to use the appropriate size of I/O unit.

8. Conclusion

This paper presents the SSD characterization algorithm to infer characteristics of SSDs that are not disclosed by the vendors, such as internal parallelism, I/O unit, and page allocation scheme, by measuring current with an oscilloscope and high-resolution current probe.

These characterization algorithms are applied to the four real SSDs. We found the internal parallelism which is the number of dies in a flash memory package, and the I/O unit which is the read/write unit larger than a page size. From these two characteristics of SSDs, its page allocation scheme is inferred.

Internal parallelism, I/O unit, and page allocation scheme are characteristics of SSDs that are not made public by the vendors. Yet, they affect I/O performance of SSDs, which is the biggest competitive factor in SSDs. If the OS knows these characteristics, the I/O performance can be improved by file system tuning. In addition, vendors will be able to devote more efforts in developing more energy efficient SSDs.

Currently, it is possible to implement SSDs whose performance is close to the limit of interface performance by placing large-sized I/O buffers and using heavy internal parallelism; however, this implementation causes a significant level of power consumption. The required design direction of SSDs is a balance between achieving I/O performance improvement and energy efficiency.

Acknowledgments

This work is supported by IT R&D program MKE/KEIT (No. 10041608, Embedded System Software for New-memory based Smart Device), by ITRC support program (IITP-2016-H8501-16-1006) and by ICT R&D program of MSIP/IITP (R0601-15-1063, Software Platform for ICT Equipments).

References

[1] K. Takeuchi, "Highly reliable low power solid-state drives (SSDs)," Proc. IEEE International Meeting for Future of Electron Devices, Kansai, Japan, pp.1–2, May 2012.
[2] L.M. Grupp, J.D. Davis, and S. Swanson, "The bleak future of NAND flash memory," Proc. 10th USENIX Conf. on File and Storage Technologies, San Jose, CA, USA, no.2, pp.1–8, Feb. 2012.
[3] R.E. Frickey III and J.M. Hughes, "Method and system to improve the performance of a multi-level cell (MLC) NAND flash memory," US Patent, pp.1–8, Dec. 2012.
[4] Y. Koh, "NAND Flash Scaling Beyond 20nm," Proc. 9th IEEE International Memory Workshop, Monterey, CA, USA, pp.1–3, May 2009.
[5] E. Yaakobi, L. Grupp, P.H. Siegel, S. Swanson, and J.K. Wolf, "Characterization and error-correcting codes for TLC flash memories," Proc. 1st IEEE International Conf. on Computing, Networking and Communications, Maui, Hawaii, USA, pp.486–491, Jan. 2012.
[6] N. Agrawal, V. Prabhakaran, T. Wobber, J.D. Davis, M.S. Manasse, and R. Panigrahy, "Design tradeoffs for SSD performance," Proc. 14th USENIX Annual Technical Conf., Boston, Massachusetts, USA, pp.57–70, June 2008.
[7] E. Seo, S.Y. Park, and B. Urgaonkar, "Empirical analysis on energy efficiency of flash-based SSDs," Proc. 1st USENIX Workshop on Power Aware Computing and Systems, San Diego, CA, USA, pp.1–5, Dec. 2008.
[8] E.D. Rejowski, P. Mordente Sr, M.F. Pillis, and T. Casserly, "Application of DLC Coating in Cylinder Liners for Friction Reduction," Proc. SAE Technical Paper, April 2012.
[9] W. Butler, Huang, H. Yao, and R. Lloyd Jr, "Disk drive including surface coated disk clamp screws with reduced coefficient of friction for mitigating disk clamp movement," US Patent, pp.1–8, Oct. 2002.
[10] S.W. Schlosser, W. Steven W, J. Schindler, S. Papadomanolakis, M. Shao, A. Ailamaki, C. Faloutsos, and G.R. Ganger, "On Multidimensional Data and Modern Disks," Proc. 4th USENIX Conf. on File and Storage Technologies, San Francisco, CA, USA, no.1, pp.1–14, Dec. 2005.
[11] J. Gim and Y. Won, "Extract and infer quickly: Obtaining sector geometry of modern hard disk drives," Proc. ACM Trans. on Storage, vol.6, no.2, pp.1–26, July 2010.
[12] O. Mesut and N. Lambert, "HDD characterization for A/V streaming applications," Proc. IEEE Trans. on Consumer Electronics, vol.48, no.3, pp.802–807, Aug. 2002.
[13] N. Talagala, R. Arpaci-Dusseau, and D. Patterson, "Microbenchmark-based Extraction of Local and Global Disk Characteristics," Proc. University of California at Berkeley, CA, USA, no.UCB/CSD-99-1063, pp.1–26, 1999.
[14] J. Schindler, J.L. Griffin, C.R. Lumb, and G.R. Ganger, "Track-aligned Extents: Matching Access Patterns to Disk Drive Characteristics," Proc. 1st USENIX Conf. on File and Storage Technologies, Monterey, CA, USA, vol.2, pp.259–274, Jan. 2002.
[15] B. Yoo, Y. Won, J. Choi, S. Yoon, S. Cho, and S. Kang, "SSD characterization: from energy consumption's perspective," Proc. USENIX 3rd Conf. on Hot topics in storage and file systems, Portland, OR, USA, p.3, June 2011.
[16] O. Mesut and N. Lambert, "HDD characterization for A/V streaming applications," Proc. IEEE Trans. on Consumer Electronics, vol.48, no.3, pp.802–807, Aug. 2002.
[17] R. Pitchumani, A. Hospodor, A. Amer, Y. Kang, E.L. Miller, and D.D.E. Long, "Emulating a Shingled Write Disk," Proc. 20th IEEE International Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems, Arlington, Virginia, USA, pp.339–346, Aug. 2012.
[18] S. Park and H. Shin, "Rigorous Modeling of Disk Performance for Real-Time Applications," Proc. 9th International Conf. on Real-Time and Embedded Computing Systems and Applications, Tainan City, Taiwan, pp.486–498, Feb. 2003.
[19] Y. Hu, H. Jiang, D. Feng, L. Tian, H. Luo, and S. Zhang, "Performance impact and interplay of SSD parallelism through advanced commands, allocation strategy and data granularity," Proc. International Conf. on Supercomputing, Tucson, Arizona, USA, pp.96–107, May 2011.
[20] T. Xie and J. Koshia, "Boosting random write performance for enterprise flash storage systems," Proc. IEEE 27th Symposium on Mass Storage Systems and Technologies, Denver, Colorado, USA, pp.1–10, May 2011.
[21] S. Boboila and P. Desnoyers, "Performance models of flash-based solid-state drives for real workloads," Proc. IEEE 27th Symposium on Mass Storage Systems and Technologies, Denver, Colorado, USA, pp.1–6, May 2011.
[22] Y. Kim, B. Tauras, A. Gupta, and B. Urgaonkar, "FlashSim: A simulator for NAND flash-based solid-state drives," Proc. IEEE 1st International Conference on Advances in System Simulation, Porto, Portugal, pp.125–131, Sept. 2009.
[23] L. Bouganim, B. Jónsson, and P. Bonnet, "uFLIP: Understanding flash IO patterns," Proc. 4th Biennial Conf. on Innovative Data Systems Research, Asilomar, CA, USA, Jan. 2009.
[24] Y.J. Seong, E.H. Nam, J.H. Yoon, H. Kim, J.-Y. Choi, S. Lee, Y.H.

Bae, J. Lee, Y. Cho, and S.L. Min, "Hydra: A Block-Mapped Parallel Flash Memory Solid-State Disk Architecture," Proc. IEEE Trans. on Computers, vol.59, no.7, pp.905–921, July 2010.

[25] V. Mohan, S. Gurumurthi, and M.R. Stan, "FlashPower: A Detailed Power Model for NAND Flash Memory," Proc. of DATE, Dresden, Germany, no.6, pp.502–507, March 2010.

[26] L.M. Grupp, A.M. Caulfield, J. Coburn, S. Swanson, E. Yaakobi, P.H. Siegel, and J.K. Wolf, "Characterizing flash memory: Anomalies, observations, and applications," Proc. ACM 42nd Internal Symposium on MICRO-42, New York, New York, USA, pp.24–33, Dec. 2009.

[27] C. Park, P. Talawar, D. Won, M.J. Jung, J.B. Im, S. Kim, and Y. Choi, "A high performance controller for NAND flash-based solid state disk (NSSD)," Proc. IEEE 21th Non-Volatile Semiconductor Memory Workshop, pp.17–20, Feb. 2006.

[28] G. Wu, X. He, and B. Eckart, "An adaptive write buffer management scheme for flash-based ssds" Proc. ACM Trans. on Storage, vol.8, no.1, pp.1–24, Feb. 2012.

[29] SAMASUNG Electronics, "2g x 8 bit/4g x 8 bit nand flash memory (k9xxg08uxm)," 2006.

[30] Intel, "Intel MD332B NAND Flash Memory Specification," 2009.

[31] F. Chen, R. Lee, and X. Zhang, "Essential roles of exploiting internal parallelism of flash memory based solid state drives in high-speed data Processing," Proc. IEEE 17th International Symposium on High Performance Computer Architecture, San Antonio, Texas, USA, pp.266–277, Feb. 2011.

[32] S. Lee, D. Shin, Y.-J. Kim, and J. Kim, "LAST: locality-aware sector translation for NAND flash memory-based storage systems," Proc. ACM SIGOPS Operating Systems Review, vol.42, no.6, pp.36–42, Oct. 2008.

[33] J. Kim, J.M. Kim, S.H. Noh, S.L. Min, and Y. Cho, "A space-efficient flash translation layer for CompactFlash systems," Proc. IEEE Trans. on Consumer Electronics, vol.48, no.2, pp.366–375, May 2002.

[34] S.-W. Lee, W.-K. Choi, and D.-J. Park, "FAST: An efficient flash translation layer for flash memory," Proc. Emerging Directions in Embedded and Ubiquitous Computing, vol.4097, pp.879–887, Springer Berlin Heidelberg, Berlin, 2006.

[35] W. Bux and I. Iliadis, "Performance of greedy garbage collection in flash-based solid-state drives," Proc. Performance Evaluation, vol.67, no.11, pp.1172–1186, 2010.

[36] B. Peleato, R. Agarwal, and J. Cioffi, "On the distribution of valid pages with greedy garbage collection for NAND flash," Proc. IEEE Statistical Signal Processing Workshop, Ann Arbor, Michigan, USA, pp.500–503, Aug. 2012.

[37] M. Murugan and D.H.C. Du, "Rejuvenator: A static wear leveling algorithm for NAND flash memory with minimized overhead," Proc. IEEE 27th Symposium on Mass Storage Systems and Technologies, Denver, Colorado, USA, pp.1–12, May 2011.

[38] L.-P. Chang, "On efficient wear leveling for large-scale flash-memory storage systems," Proc. ACM symposium on Applied computing, Seoul, Korea, pp.1126–1130, March 2007.

[39] C. Dirik and B. Jacob, "The performance of PC solid-state disks (SSDs) as a function of bandwidth, concurrency, device architecture, and system organization," Proc. ACM SIGARCH Computer Architecture News, vol.37, no.3, pp.279–289, June 2009.

[40] M. Wei, L.M. Grupp, F.E. Spada, and S. Swanson, "Reliably erasing data from flash-based solid state drives," Proc. USENIX 9th Conf. on File and Stroage Technologies, San Jose, California, USA, pp.8–8, Feb. 2011.

[41] Linux Journal Staff, "upFront," Linux J., vol.2005, no.129, p.22, Jan. 2005.

[42] H.-T. Lue, T.-H. Hsu, S.-Y. Wang, E.-K. Lai, K.-Y. Hsieh, R. Liu, and C.-Y. Lu, "study of incremental step pulse programming (ISPP) and STI edge effect of BE-SONOS NAND Flash," Proc. 46th IEEE International Symposium on Reliability Physics, Phoenix, Arizona, USA, pp.693–694, April 2008.

[43] G. Hong, "Analysis of peak current consumption for large-scale, parallel flash memory," Proc. Workshop for Operating System Support for Non-Volatile RAM, Jeju, Korea, April 2011.

Balgeun Yoo received the BS degree in Computer Engineering from Dankook University in 2008. He is currently working towards his PhD degree in the Division of Computer Science and Engineering at Hanyang University, Seoul, Korea. His research interests include deduplication systems and operating systems.

Seongjin Lee is a PostDoc. at Embedded Software Systems Laboratory in Department of Computer Software at Hanyang University, Seoul Korea. He did his BS and MS in Department of Electronics and Computer Engineering, Hanyang University, Seoul Korea in 2006 and 2008, respectively. He received his Ph. D in Computer Engineering in the same university in 2015. His research interests include system performance, measurements, analysis, characterization, and classification.

Youjip Won is currently Professor at Division of Electrical and Computer Engineering, Hanyang University, Seoul Korea. He is leading Embedded Software System Lab. He did his BS and MS in Dept. of Computer Science, Seoul National University, Seoul, Korea in 1990 and 1992, respectively. He received his Ph. D in Computer Science from University of Minnesota in 1997. Before joining Hanyang University in 1999, he worked at Intel Corp. as Server Performance Analyst. His research interests include Network Traffic Modeling, Analysis and Characterization, Multimedia system and networking, File and Storage subsystem, Lower power Storage System. In 2006, Multimedia File System Project funded by Samsung Electronics was awarded "Best Academy-Industry Collaboration Practice in Samsung Electronics". In 2007, he was awarded "National Research Lab" grant which is highly selective and prestigeous governmental grant.