Design Tradeoffs of SSDs: From Energy Consumption's Perspective

SEOKHEI CHO, CHANGHYUN PARK, YOUJIP WON, SOOYONG KANG, and JAEHYUK CHA, Hanyang University
SUNGROH YOON, Seoul National University
JONGMOO CHOI, Dankook University

In this work, we studied the energy consumption characteristics of various SSD design parameters. We developed an accurate energy consumption model for SSDs that computes aggregate, as well as component-specific, energy consumption of SSDs in sub-msec time scale. In our study, we used five different FTLs (page mapping, DFTL, block mapping, and two different hybrid mappings) and four different channel configurations (two, four, eight, and 16 channels) under seven different workloads (from large-scale enterprise systems to small-scale desktop applications) in a combinatorial manner. For each combination of the aforementioned parameters, we examined the energy consumption for individual hardware components of an SSD (microcontroller, DRAM, NAND flash, and host interface). The following are some of our findings. First, DFTL is the most energy-efficient address-mapping scheme among the five FTLs we tested due to its good write amplification and small DRAM footprint. Second, a significant fraction of energy is being consumed by idle flash chips waiting for the completion of NAND operations in the other channels. FTL should be designed to fully exploit the internal parallelism so that energy consumption by idle chips is minimized. Third, as a means to increase the internal parallelism, increasing way parallelism (the number of flash chips in a channel) is more effective than increasing channel parallelism in terms of peak energy consumption, performance, and hardware complexity. Fourth, in designing high-performance and energy-efficient SSDs, channel switching delay, way switching delay, and page write latency need to be incorporated in an integrated manner to determine the optimal configuration of internal parallelism.

Categories and Subject Descriptors: B.3.3 [**Memory Structures**]: Performance Analysis and Design Aids

General Terms: Design, Measurement

Additional Key Words and Phrases: SSD, NAND flash, energy consumption, FTL, parallelism, simulator

ACM Reference Format:
Seokhei Cho, Changhyun Park, Youjip Won, Sooyong Kang, Jaehyuk Cha, Sungroh Yoon, and Jongmoo Choi. 2015. Design tradeoffs of SSDs: From energy consumption's perspective. ACM Trans. Storage 11, 2, Article 8 (March 2015), 24 pages.
DOI: http://dx.doi.org/10.1145/2644818

This work is sponsored by IT R&D program MKE/KEIT [No.10035202, Large Scale hyper-MLC SSD Technology Development] and by the MSIP (Ministry of Science, ICT & Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (NIPA-2014- H0301-14-1017) supervised by the NIPA (National IT Industry Promotion Agency).
Authors' addresses: S. Cho and C. Park, Department of Computer Software, Hanyang University, 222 Wangsimni-ro, Seongdong-gu, Seoul 133-791, Korea; emails: {misost, pch1984}@hanyang.ac.kr; Y. Won (corresponding author), S. Kang, and J. Cha, Division of Computer Science and Engineering, Hanyang University; emails: {yjwon, sykang, chajh}@hanyang.ac.kr; S. Yoon, Department of Electrical and Computer Engineering, Seoul National University, 1 Gwanak-ro, Gwanak-gu, Seoul 151-742, Korea; email: sryoon@snu.ac.kr; J. Choi, Department of Software, Dankook University, 152, Jukjeon-ro, Suji-gu, Yongin-si, Gyeonggi-do, 448-701, Korea; email: choijm@dankook.ac.kr.

DOI: http://dx.doi.org/10.1145/2644818

1. INTRODUCTION

Due to the rapid advancement of flash memory technology (e.g., adoption of sub-20nm process technology and multiple bits per cell), storage density of NAND flash-based storage devices has improved significantly. This has led to the decrease in cost/GB and subsequently accelerated the wider deployment of NAND flash-based storage devices. In 2012, for the first time ever in the history of the semiconductor, the global sales of NAND flash exceeded that of DRAM [IC-insights 2012]. The NAND flash-based storage device positions itself as a mainline storage system not only in mobile devices (e.g., smartphones and smart pads) but also in desktop PCs, notebooks, and enterprise servers [Narayanan et al. 2009; Intel 2012]. NAND flash-based storage devices exhibit superior physical characteristics to HDDs in terms of noise, heat, and shock resistance. From the performance point of view, it exhibits 20× improvement in random IO performance and shorter I/O latency. A Solid-State Drive (SSD) positions itself as a tier for hybrid storage systems for large-scale distributed systems [Grider 2011; Strande et al. 2012] or as a storage system for checkpointing the memory snapshot of a supercomputer [He et al. 2010; Ni et al. 2012].

Building an energy-efficient computer system is of critical concern in the high-performance computing community [Frachtenberg et al. 2011; Pillai et al. 2012; Tsirogiannis et al. 2010]. SSDs have been widely perceived as a means to deliver energy-efficient computer systems, replacing HDD-based storage systems [Poess and Nambiar 2010]. Recently, SSD vendors have adopted aggressive internal parallelism to boost the I/O performance of SSDs. Most SSDs adopt eight channels, and each channel has one or two flash packages. X25M from Intel [Intel 2009b], for example, can perform 20 page-write operations simultaneously. The energy consumption rate of modern SSDs is much denser than that of legacy HDDs from the J/sec and J/cm^3 point of view. A single HDD can consume as much as 8W. On the other hand, the peak power consumption of an SSD is subject to the number of flash dies, which can be programmed in parallel and can be as high as 16W.

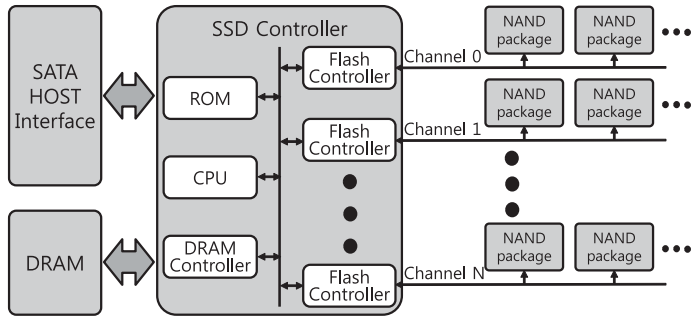
Therefore, it is critical that every design choice of a modern SSD is carefully examined from the aspect of energy consumption. In this work, we dedicate our effort to understanding the energy consumption characteristics of various SSD components: address mapping, garbage collection, internal parallelism, and page size. For our study, we developed energy consumption models for SSD components: flash memory, microcontroller, interface, and main memory. Our model is not a full-fledged energy consumption simulator, but the physical experiment shows that our simulator is within 8% error from physical SSDs in terms of energy consumption. We implemented our energy consumption model on top of an existing DiskSim-based SSD simulator [Agrawal et al. 2008].

The rest of the article is organized as follows: Section 2 describes the basics of NAND flash, SSDs, and energy consumption. Section 3 explains the energy consumption model for SSDs. Section 4 describes the organization of our simulator, *Energysim*. Section 5 contains the results of our case study, and Section 6 contains related work. Section 7 concludes the article.

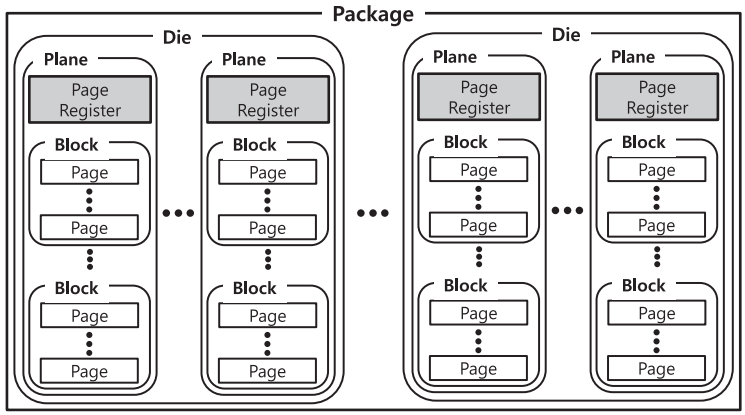
2. MODERN SSDS AND ENERGY CONSUMPTION

2.1. SSD Organization

Figure 1 shows the overall architecture of an SSD. It consists of the SSD controller, RAM (DRAM or SRAM), NAND, and the interfaces. The SSD controller manages I/O operations and runs firmware (e.g., FTL, buffer manager, and garbage collector). DRAM or SRAM is typically used as the internal device cache and has two purposes: data buffering and map caching [Shim et al. 2010]. The interfaces refer to the host interface



(a) SSD Controller



(b) NAND Package

Fig. 1. SSD hardware diagram.

and flash interface. The host interface connects the SSD and the host system. The flash interface connects the SSD controller and the NAND chips and transfers data between the controller and the page register. The speed of the flash interface varies subject to the specific NAND flash interface (e.g., ONFI [ONFI 2011] and toggle NAND).

SSDs organize flash memory packages in channels and ways. Each channel has its own page register, which acts as a buffer between the NAND flash chip and NAND flash controller. Flash pages in different channels can be accessed in parallel. The accesses to flash pages that are in the same channel but in different flash packages are interleaved, which means that a following operation needs to wait until the preceding operation releases the page register.

The degree of parallelism is usually governed by the number of channels multiplied by the number of ways (the number of flash packages in a channel). If a flash package consists of multiple planes, the maximum degree of parallelism is obtained by (# of channels) $\times$ (# of ways in a channel) $\times$ (# of planes in a way). One should determine an optimal combination of the number of channels and the number of ways in achieving a given degree of parallelism. One of the important factors that governs the optimal configuration is peak current consumption. Figure 2 schematically illustrates the time it took to write four pages and the level of current it consumed to complete the task under two different SSD configurations. In the first configuration, the SSD writes four pages to four channels, one page in each channel (Figure 2(a)). In the second configuration, the SSD writes four pages to two channels, two pages in each channel

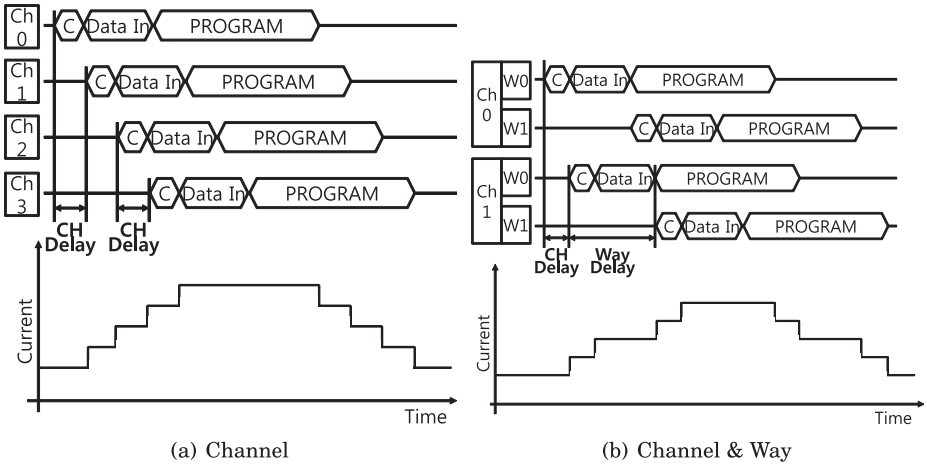


Fig. 2. Channel and way timing diagrams and current consumption.

Table I. HDD and SSD Specifications

Model	Capacity	Read/Write	# of Channel	Released Year
Caviar[1]	80GB	150/94MB/s	-	2005
X25M[2]	80GB	250/70MB/s	10	2008
MXP[3]	128GB	220/200MB/s	8	2009
Vertex 1[4]	60GB	230/130MB/s	8	2009
OCTANE[4]	512GB	535/400MB/s	8	2011
840[3]	250GB	540/250MB/s	8	2012

[1]WD; [2]Intel; [3]Samsung; [4]OCZ

(Figure 2(b)). Each of the two writes in a channel goes into different chips, exploiting way parallelism. In the first configuration, since the channels can transfer data in an independent manner, four pages in each of the four channels can be written in parallel and the write operations of all NAND flashes are mostly overlapped. Channel switch delay denotes the interval between the start times of two consecutive flash pages when they are written on the two flash chips that are in the adjacent channels. Way switch delay denotes the time interval between the start times of two consecutive flash pages when they are written on two different flash pages that are attached to the same channel. In multichannel and multiway SSDs, switching channels and ways accompanies channel switch delay and way switch delay. There are a number of ways to achieve the same degree of parallelism. For example, we can adopt either four-channel one-way or two-channel two-way configurations to achieve a parallelism degree of 4. Using a smaller number of channels ameliorates the stress caused by peak current, that is, energy consumption, heat, channel interference, and so forth.

2.2. Energy Consumption of SSDs

We examined the current consumption behavior of one HDD and five commercially available SSDs (Table I). Some SSDs exhibited higher peak current consumption than the HDD. We used the current probe[1](5V) and captured the current consumption. The sampling interval was 100μsec. The graphs in Figure 3 show the averages of 10,000 samples. The energy consumption for every 100μsec time slot can be computed as

[1]Tektronix TCP202.

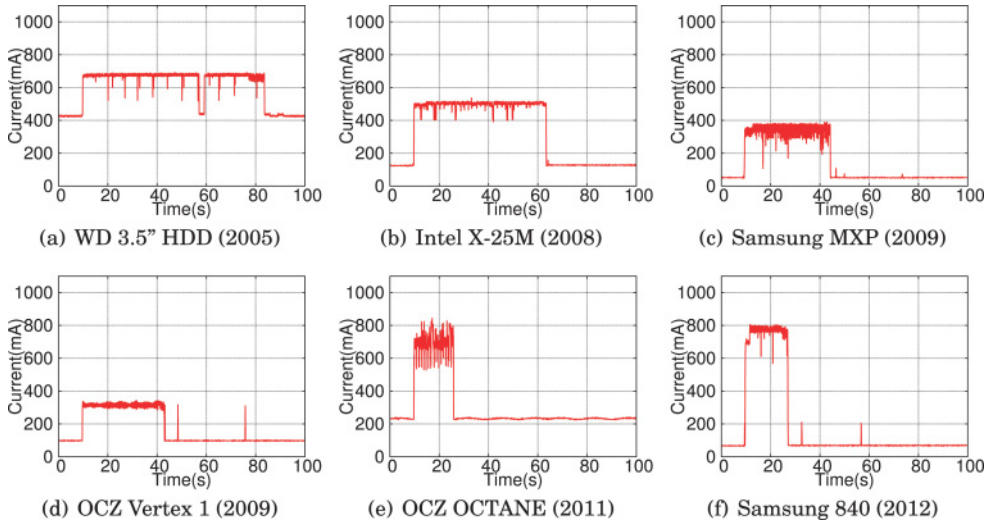


Fig. 3. Current consumption of sequential write operation (voltage: 5V, sampling interval: 100μsec).

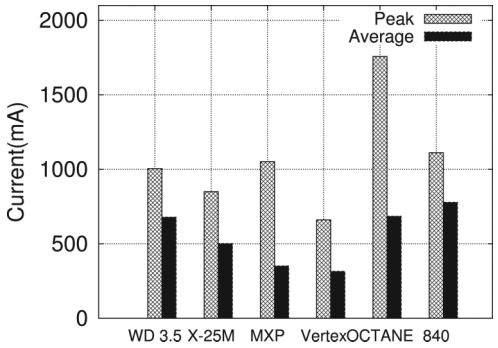


Fig. 4. Peak and average (active state) current.

(sampling interval) $\times$ (current) $\times$ (voltage). The workload is 4GB sequential write operation. Figure 3 illustrates the time series of the current consumption. To easily compare the performance and energy consumption, we used the same x- and y-scales in all graphs. For the 3.5" HDD, the current consumption level stayed at 690mA when active. The early SSD models consumed less energy than the hard disk drive did (Figure 3(c) and Figure 3(d)). The recently released SSDs (Figure 3(e) and Figure 3(f)) consumed approximately 800mA when active. These two SSDs consumed 15% more current than the HDD at its peak. The peak energy consumption is another important characteristic of the storage device because the controller circuitry needs to be built to sustain the peak current. Otherwise, the SSD controller may be subject to interference, ground bounce, blackouts, and so forth. Figure 4 illustrates the average and peak energy consumptions of the five SSDs and one HDD. We can see that the peak currents (mA) of recently released SSDs are much higher than those of the legacy HDD.

3. MODELING THE ENERGY CONSUMPTION OF SSD COMPONENTS

We developed an energy consumption model for each component of an SSD: the SSD controller, DRAM, NAND flash, and host interface. Individual components are assumed

Table II. Variables of Equations

Name	Description
E^{con}, E^{dr}, E^{fl}, E^{bus}	The total energy consumption by the SSD controller, DRAM, NAND flash, and bus
E^{con}_{active}, E^{dr}_{active}, E^{fl}_{active}	Active-state energy consumption by the SSD controller, DRAM, and NAND flash
E^{con}_{idle}, E^{dr}_{idle}, E^{fl}_{idle}	Idle-state energy consumption by the SSD controller, DRAM, and NAND flash
I^{con}_{active}	Active-state current consumption by the SSD controller
I^{con}_{idle}, I^{dr}_{idle}, I^{fl}_{idle}	Idle-state current consumption by the SSD controller, DRAM, and NAND flash
I^{bus}	Current consumption by bus
T_{total}	The total simulation time
T^{con}_{idle}, T^{dr}_{idle}, T^{fl}_{idle}	The total idle time for the SSD controller, DRAM, and NAND flash
T^{bus}	The time spent in bus
V^{con}, V^{dr}, V^{fl}, V^{bus}[2]	Applied voltage in the SSD controller, DRAM, NAND flash, and bus
E^{dr}_{OP}	The energy consumption for processing one operation in DRAM
I^{dr}_{OP}	Current consumption for processing one operation in DRAM. The current consumption is the same for read and write operations.
N^{dr}_{OP}	The total number of operations in DRAM
T^{dr}_{OP}	The time to perform the operation in DRAM
E^{fl}_{read},E^{fl}_{write},E^{fl}_{erase}	The energy consumption by NAND flash for read, write, and erase operations
I^{fl}_{read},I^{fl}_{write},I^{fl}_{erase}	Current consumption by NAND flash for read, write, and erase operations
N^{fl}_{read},N^{fl}_{write},N^{fl}_{erase}	The number of read, write, and erase operations in NAND flash
T^{fl}_{read},T^{fl}_{write},T^{fl}_{erase}	The time spent for read, write, and erase operations in NAND flash

con: SSD controller; dr: DRAM; fl: NAND flash; bus: host Interface.

to be in one of the three states: active, idle, or power-off. Only NAND flash and host interface can be in a power-off state. The active state of NAND flash is further divided into read, write, and erase substates. Energy is computed via (average current for a given state) $\times$ (duration) $\times$ (voltage).

3.1. SSD Controller

In this work, most of our efforts are focused on investigating the energy consumption behavior of NAND flash devices under varying channel/way configurations and FTL algorithms. Less attention has been paid to accurately modeling the energy consumption of the rest of the components (e.g., SSD controller and DRAM). Detailed modeling of these components is beyond the scope of this work. The SSD controller continues to consume energy to maintain its internal components' stand-by even in the idle state. In the active state, the SSD controller is processing I/O requests from the host or is performing internal I/O operations such as garbage collection. The SSD controller consumes more energy in the active state than in the idle state, and the level of energy consumption varies depending on the type of command executed. In the idle state, the energy consumption of an SSD controller stays almost constant. The active-state energy consumption varies widely subject to the amount of data transferred and to the number of active channels and ways involved in the transfer. In our model, the SSD controller is in an active state if one of the following three conditions is met: (1) the SSD

[2]We assigned separate variables for the voltages for DRAM, flash, and bus to represent the case where each of these has different values.

controller is processing an I/O command from the host, (2) the SSD controller is performing garbage collection, or (3) the SSD controller is performing buffer management. Otherwise, the SSD is assumed to be in an idle state.

$$E^{con}_{active} = \left(I^{con}_{active} \cdot V^{con}\right) \cdot \left(T_{total} - T^{con}_{idle}\right) \tag{1}$$

$$E^{con}_{idle} = \left(I^{con}_{idle} \cdot V^{con}\right) \cdot T^{con}_{idle} \tag{2}$$

$$E^{con} = E^{con}_{active} + E^{con}_{idle} \tag{3}$$

Equations (1), (2), and (3) represent the energy consumptions in the active state, in the idle state, and in total, respectively. E^{con}_{active} is the amount of energy the SSD controller consumes in the active state. I^{con}_{active} is the cumulative current consumed in the active state, and V^{con} is the voltage applied to the controller. The time spent in the active state is obtained by subtracting the idle time, T^{con}_{idle}, from the total simulation time, T_{total}. E^{con}_{idle} is the energy consumed by the SSD controller in the idle state, and I^{con}_{idle} is the consumed current in the idle state. The total energy consumed by the SSD controller is denoted by E^{con} and is obtained by adding up the energy consumed in active and idle states.

3.2. DRAM

DRAM is in the active state when it is reading or writing. Otherwise, it is in the idle state. We assume that the read and write operations consume the same amount of current.

$$E^{dr}_{OP} = \left(I^{dr}_{OP} \cdot V^{dr}\right) \cdot T^{dr}_{OP} \tag{4}$$

$$E^{dr}_{active} = E^{dr}_{OP} \cdot N^{dr}_{OP} \tag{5}$$

$$E^{dr}_{idle} = \left(I^{dr}_{idle} \cdot V^{dr}\right) \cdot T^{dr}_{idle} \tag{6}$$

$$E^{dr} = E^{dr}_{active} + E^{dr}_{idle} \tag{7}$$

Equation (4) shows the amount of energy consumed for processing one operation in DRAM. I^{dr}_{OP} is the amount of current required for DRAM operation. V^{dr} is the voltage applied to DRAM, and T^{dr}_{OP} is the time for DRAM to perform the operations. Equation (5) is the total energy DRAM consumes in the active state. N^{dr}_{OP} is the total number of operations. Equation (6) represents the energy DRAM consumes in the idle state. $T^{dr}_{idle} = T_{total} - (N^{dr}_{OP} \cdot T^{dr}_{OP})$ and corresponds to the total simulation time minus the operation time. Since DRAM is kept powered on for preserving data until the SSD is powered off, the idle time corresponds to the total simulation time minus the operation time. Equation (7) is the total energy consumption by DRAM, which is the sum of the energy consumption in the idle and active states.

3.3. NAND Flash

A single NAND chip normally consumes 20 to 30mA to perform read, program, or erase operations [Grupp et al. 2009]. In the idle state, an SSD consumes energy for precharging bit lines for NAND operations and for exchanging clock signals with the flash controller in the SSD controller. The active state of NAND flash is divided into three substates: read, write, or erase. According to the datasheet by an NAND flash manufacturer [Samsung 2012], NAND flash consumes almost an identical amount of energy regardless of the type of operation. However, according to Grupp et al. [2009], this is not realistic, and different operations consume different amounts of energy. In this article, we assume different current consumption levels for each NAND operation.

Note that NAND flash is nonvolatile, and we can turn it off selectively if needed. The energy consumption of a single NAND flash die can be computed as follows:

$$E^{fl}_{read} = \left(I^{fl}_{read} \cdot V^{fl}\right) \cdot T^{fl}_{read} \tag{8}$$

$$E^{fl}_{write} = \left(I^{fl}_{write} \cdot V^{fl}\right) \cdot T^{fl}_{write} \tag{9}$$

$$E^{fl}_{erase} = \left(I^{fl}_{erase} \cdot V^{fl}\right) \cdot T^{fl}_{erase} \tag{10}$$

$$E^{fl}_{active} = E^{fl}_{read} \cdot N^{fl}_{read} + E^{fl}_{write} \cdot N^{fl}_{write} + E^{fl}_{erase} \cdot N^{fl}_{erase} \tag{11}$$

$$E^{fl}_{idle} = \left(I^{fl}_{idle} \cdot V^{fl}\right) \cdot T^{fl}_{idle} \tag{12}$$

$$E^{fl} = E^{fl}_{active} + E^{fl}_{idle}. \tag{13}$$

Equations (8), (9), and (10) represent the amount of energy consumed by NAND flash for read, write, and erase operations, respectively. I^{fl}_{read}, I^{fl}_{write}, and I^{fl}_{erase} are current consumptions for read, write, and erase operations, respectively. T^{fl}_{read}, T^{fl}_{write}, and T^{fl}_{erase} are the time spent for read, write, and erase operations, respectively. V^{fl} is the voltage applied to the flash memory. We calculated the total energy consumption of operations by adding up the energy consumption of each state. Equation (11) shows the energy consumed by NAND flash in the active state, which is the sum of the energy consumed by each type of operation. N^{fl}_{read}, N^{fl}_{write}, and N^{fl}_{erase} are the number of read, write, and erase operations, respectively. Equation (12) indicates the energy consumed by NAND flash in the idle state. T^{fl}_{idle} is the idle time of NAND flash and is given by $T_{total} - (T^{fl}_{read} \cdot N^{fl}_{read} + T^{fl}_{write} \cdot N^{fl}_{write} + T^{fl}_{erase} \cdot N^{fl}_{erase})$. Because it is possible to power off NAND flash as needed, we can subtract the power-off time from the total time. The proposed simulator provides two types of energy consumption in the idle state: One is the energy consumed in the idle state without any power-off state. The other is the energy consumed in the idle state under the assumption that NAND flash is turned off when there is no I/O request. Equation (13) is the total energy consumption by NAND flash, which is the sum of the energy consumptions in active and idle states.

3.4. Host Interface

In SSDs, according to Grupp et al. [2009], the energy involved in data transfers is approximately three times higher than that in the idle state in NAND flash. The power used in data transfers is not negligible and should be considered in modeling the SSD energy consumption. The energy used by a data bus is less than that of the SSD controller, DRAM, or NAND flash in the active state. Energysim considers only the energy consumed during data transfers, ignoring the energy consumed in the idle state:

$$E^{bus} = (I^{bus} * V^{bus}) * T^{bus} \tag{14}$$

I_{bus} and V_{bus} are the current and the voltage applied to the bus, respectively, and T_{op} is the time for data transfer through the bus.

4. SIMULATOR

4.1. Simulator Design

We modified the existing trace-driven simulator model for an SSD [Agrawal et al. 2008] to study its energy consumption behavior. We modified this simulator to support multichannel and multiway configurations: the channel switch delay and way switch delay can be adjusted via input parameters. We call the simulator *Energysim*.[3] Figure 5

[3]The simulator is publicly available at https://github.com/ESOS-Lab/EnergySim/.

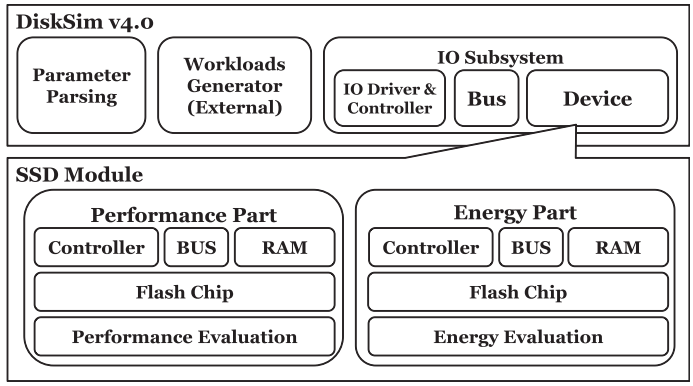


Fig. 5. Simulator system overview.

illustrates the structure of our simulator. *Energysim* models the energy consumption of the SSD controller, DRAM, NAND flash, and host interface based on the model described in Section 3. It generates the aggregate energy consumption for a given period of time as well as the energy consumption for every interval between the events. Also, Energysim generates the energy consumption statistics for each type of NAND operations.

The existing DiskSim model for SSDs [Agrawal et al. 2008] has only page mapping [Ban 1995]. We implemented DFTL [Gupta et al. 2009], block mapping [Ban 1999], and BAST [Jesung et al. 2002]. DFTL uses page mapping with caching and demand paging for mapping table management. In block mapping, consecutive sectors are relocated to different flash blocks when some of the sectors are updated. In page mapping, each updated sector can be relocated into any NAND flash pages. Our block mapping is based on Kuo et al. [2008]. BAST (Block Associative Sector Translation) categorizes NAND flash blocks into two categories: data block and log block. The data block and log block are managed by block mapping and page mapping, respectively. An incoming write request is first written to the log block. When there are no pages available in the log block, the valid pages in the log block are consolidated with the pages in the data block and the respective log block is erased. This process is called *log block cleaning*. For log block cleaning, BAST employs full merge, switch merge, and replacement. Replacement is an operation that erases victim log block and allocates a new free block. Among the three types of log block cleaning operations, we found that the replacement operation causes too many erase operations and negatively affects the performance. In this work, we devised modified BAST, which does not perform the replacement operation: we named this modified version BAST*. When the erase count of the victim block exceeds the average erase count by 15% or more, the FTL switches the contents of the victim block with those of a cold block. A cold block is a block that has not been erased for the longest period of time [Gal and Toledo 2005].

In real SSDs, there exists a delay in switching channels or ways [Yoo et al. 2011]. The SSD simulator by Agrawal et al. does not incorporate this characteristic and thus cannot accurately simulate the real-world behavior of an SSD. Our simulator models the channel-switch and way-switch delays. The delay information is supplied as a configuration parameter.

4.2. Simulator Validation

We validated the accuracy of our simulator against a commercially available SSD model, Intel X25M. We used a current probe, Tektronix TCP202, and the oscilloscope, Tektronix DPO 3012, to capture the current consumption behavior of X25M. The total

Table III. Energysim Parameters [Intel 2009b] (NAND Flash Spec. Is from Intel [2009a])

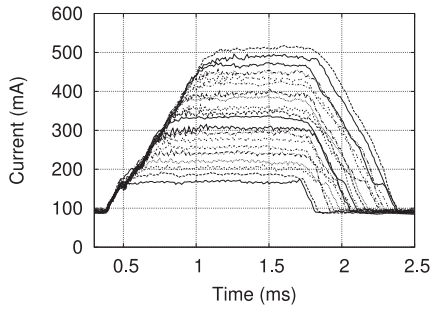
Read/Write/Erase	50μs/500, 900μs/2.0ms
Register (1 byte)	20ns
Page Size	4KB
Pages per Block	128
Blocks per Plane	2,048
Planes per Package	2
NAND Energy Specifications	
Read/Program/Erase	20mA
Idle	3mA
Voltage	3.3V
SSD Controller	
Active/Idle	30mA/15mA
DRAM (512MB,page) Active/Idle	100mA/20mA
DRAM (8MB,other) Active/Idle	77mA/3mA

energy consumption is computed by multiplying the input voltage (5V), the total current, and the execution time, that is, $E_{Drive} = V_{Drive} * I_{Drive} * Time$. In this experiment, we turned off the DRAM cache[4] of X25M to ensure the accuracy of measurement. Table III illustrates the parameters used in Energysim. These values are obtained from the SSD datasheet [Intel 2009b] and NAND flash datasheet [Intel 2009a]. In this simulation, we used different sizes of DRAM for page mapping and DFTL to examine the difference in energy consumption caused by page table accesses. In simulating page mapping, our Energysim is configured with 512Mbyte DRAM with 100mA and 20mA current consumptions for active and idle states, respectively. In other mappings (DFTL, Block, and BAST), our model is configured with 8Mbyte DRAM with 77mA and 3mA current consumption for active and idle states, respectively.

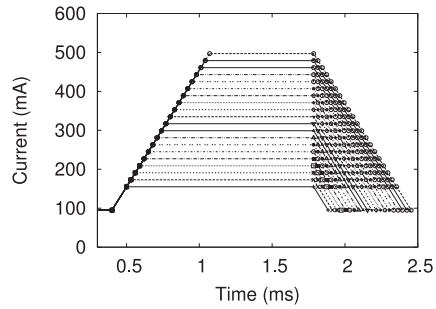
We `open()` the raw device and wrote different sizes of data ranging from 4KByte to 160KByte in increments of 4KByte. Figure 6(a) and Figure 6(c) illustrate the current consumptions of writing different sizes of blocks varying from 4KB to 160KB. We use two graphs since the current consumption behavior radically changes when the I/O size becomes larger than 80KByte. We can see in Figure 6(a) that current consumption increases by 17mA steps as we increase the data size by 4KByte. This stepwise increase continues until the I/O size reaches 80KByte. This is because X25M has a 10-channel and two-way configuration, and with an 80-KByte write, 20 page write operations are interleaved across the channels and across the ways, fully exploiting its internal parallelism. When FTL writes 80KByte, current consumption reaches the peak, 520mA.

As we can see in Figure 6(a) and Figure 6(c), as the data size increases, it takes longer for X25M to reach its maximum current consumption level. For example, when we write one page, it takes 30μsec for the SSD to reach its maximum current consumption level. When we write two pages, it takes 60μsec for the SSD to reach its peak current consumption. A possible explanation to this phenomenon is that the two pages are written to different channels and it takes 30μsec to switch the channel. When the I/O size exceeds 80KByte, it becomes a different situation. When we write 84KByte (21 pages), the first page and the 21st page are written to the same die. Therefore, the 21st write request can only start after the first write completes. If the page program (write) latency is shorter, that is, 500μsec, than the sum of all channel-switch and way-switch delays, that is, 30μsec$\times$20, the write requests will complete before the 21st write operation starts. In this case, the total current consumption continues to decrease until

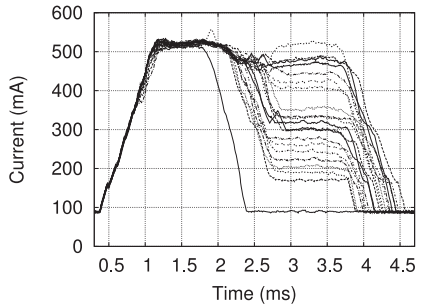
[4]SATA command 82h.



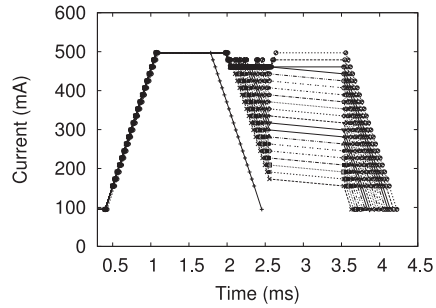
(a) Real Data (record size: 4KB–80KB, Δ=4KB)



(b) Simulation Data (record size: 4KB–80KB, Δ=4KB)



(c) Real Data (record size: 80KB–160KB, Δ=4KB)



(d) Simulation Data (record size: 80KB–160KB, Δ=4KB)

Fig. 6. Comparison between Intel X25M and Energysim on current consumption behavior of write operations (10 channels, two chips/channel, 4Gbyte/chip, $\Delta = 4$KByte).

Table IV. Energy Consumption: Real Device Versus Simulated Device (in mJ)

Write IO Size	X25-M (mJ)	Simulated X25-M (mJ)
4K	1.3	1.2 (−6.0%)
40K	2.5	2.4 (−5.8%)
80K	4.0	3.7 (−8.1%)
120K	7.3	6.7 (−8.0%)
160K	8.6	8.2 (−4.5%)

the 21st write starts. We collected the block-level trace from this experiment using blktrace and obtained energy consumption behavior from our simulator with the collected trace. Figure 6(b) and Figure 6(d) illustrate the result. Figure 6(c) illustrates current consumption of writing 84KByte to 160KByte on the real SSD (X25M). We can see that the simulator exhibits very similar current consumption patterns to the physical SSD in all these cases. Table IV compares the aggregate energy consumptions of the real SSD and the simulation model. The energy consumption of the Energysim-based SSD has an 8% worst-case relative error.

5. CASE STUDIES

We investigated the energy consumption characteristics of SSDs under a variety of options such as different address mapping, internal parallelism, and DRAM size. In this study, we used five FTLs (page level, DFTL, block level, BAST, and BAST*) and four different channel configurations (two, four, eight, and 16 channels). The latency of NAND operations and their current consumptions are set as specified in NAND flash

Table V. SSD Specifications (NAND Flash Spec. Is from Intel [b])

SSD model	
Total Capacity	460GB (512GB)
Number of Channels	2, 4, 8, 16
Number of Packages	16
Pages per Block	64, 128, 256
Page Size	4KB
Serial Access	20ns
Flash Page Read/Write/Erase	50μs/900μs/2ms

Table VI. Workload Characteristics

Workload	Read (%)	Seq. (%)	Avg. Read Size (KB)	Avg. Write Size (KB)
Financial1 [SPC 2009]	15.4	2.4	2.3	3.7
Financial2 [SPC 2009]	78.5	4.3	2.3	2.9
Homes [Koller and Rangaswami 2010]	21.7	45.6	16.7	4.0
MSNfs [Kavalanekar et al. 2008]	66.1	6.0	10.0	11.0
FileZilla [ESOSLab 2012]	0.1	62.4	10.4	436.9
Torrent [ESOSLab 2012]	34.3	8.6	14.7	36.9
GIMP [ESOSLab 2012]	48.7	17.9	136.2	15.1

memory datasheet [Intel 2009a], which are summarized in Table V. Overprovisioning degree is set to 10%.

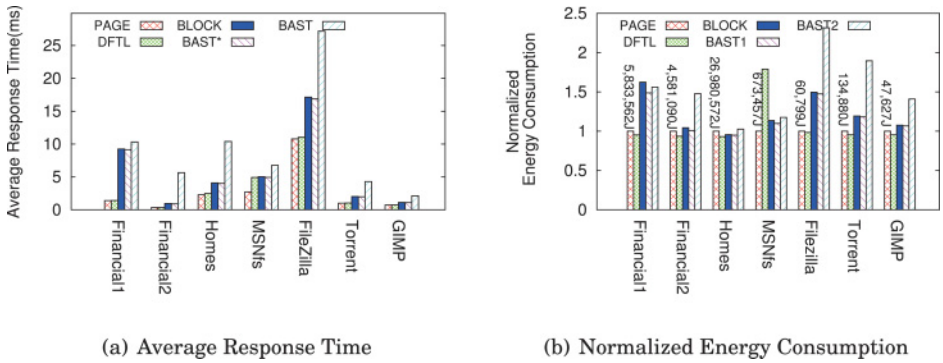
Queue depth is set to 32 in our simulator. We performed a set of experiments with different queue depths from depth 1 to depth 64. According to our experiments, the performance and the energy consumption are insensitive to the device queue depth, and we only show the result with queue depth 32 due to the space limit.

5.1. Workload Summary

Table VI shows the seven workloads used in this study. Financial1 and Financial2 are the I/O traces generated by an OLTP program in financial systems [SPC 2009]. MSNfs is the I/O trace gathered in the MSN Storage back-end file server [Kavalanekar et al. 2008]. The Homes workload is collected from the home directory in an NFS server, which consists of several research group activities such as developing, testing, technical writing, and plotting [Koller and Rangaswami 2010]. The remaining three workloads, FileZilla, Torrent, and GIMP, are from in-house traces [ESOSLab 2012].

5.2. FTL and Energy Consumption

We examined the I/O latency of the five FTLs under the seven workloads (Figure 7(a)). The page mapping yields the best performance in all workloads. DFTL exhibits 14% longer latency than page mapping. Compared with page mapping, block mapping and two hybrid mappings yield 2× or longer latencies. BAST shows the worst performance due to the replacement merge overhead. The performance differences between FTLs become more significant as the fraction of write operations increases. Figure 7(b) shows the normalized energy consumption. In most workloads, except MSNfs, DFTL is more energy efficient than page mapping. This is because DFTL has a smaller memory footprint than page mapping does since it maintains a smaller subset of mapping table in the memory. The page mapping maintains the whole mapping table in its memory. With the MSNfs workload, DFTL suffers from frequent cache misses of the map table. DFTL triggers a large number of NAND operations to synchronize the contents of map cache and the address mapping table in the NAND flash.



(a) Average Response Time (b) Normalized Energy Consumption

Fig. 7. Average response time and normalized energy consumption.

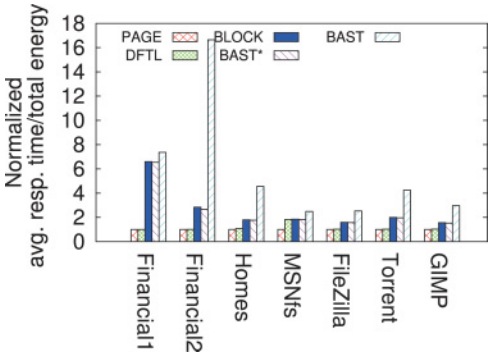


Fig. 8. Normalized average response time/total energy consumption.

To examine the energy efficiency of FTLs, we computed the (response time)/J for each FTL and normalized it against page mapping. Figure 8 illustrates the result. DFTL is the most energy-efficient FTL from the performance perspective.

We examined the energy consumption involved in accessing the mapping table in page mapping and in DFTL. In page mapping, we assumed that the entire page table is cached in DRAM. In DFTL, mapping table access occasionally triggers the NAND flash operation due to cache miss. In DFTL, energy consumption caused by mapping table access varies widely subject to its access locality. Figure 9 illustrates the result. Financial1 and Financial2 workloads are highly skewed. In DFTL, the mapping table lookup rarely triggers NAND operations since the hit ratio is quite high (specifically, the hit ratios of the two workloads are 98.5% and 97.8%, respectively). For the mapping table operation, DFTL consumes only one-quarter of the energy consumed by page mapping. In the MSNfs workload, the mapping table hit ratio is 25.3% in DFTL, causing frequent mapping table access operations on NAND flash. In this workload, DFTL consumes 11× more energy than page mapping does in the mapping table operation.

In SSDs, performance is governed by the number of NAND operations involved in servicing a given I/O request. The actual number of write operations caused by a write request from the host (e.g., SATA) command varies widely subject to the mapping algorithm, wear-leveling algorithm, garbage collection algorithm, and so forth. The ratio of the number of pages written from the host to the actual number of pages written in the flash storage is called the write amplification factor. The performance and energy consumption vary subject to the actual number of write operations in

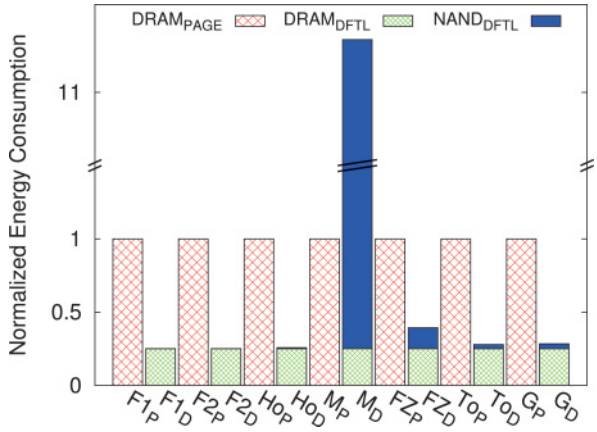


Fig. 9. Effect of mapping table caching on energy consumption.

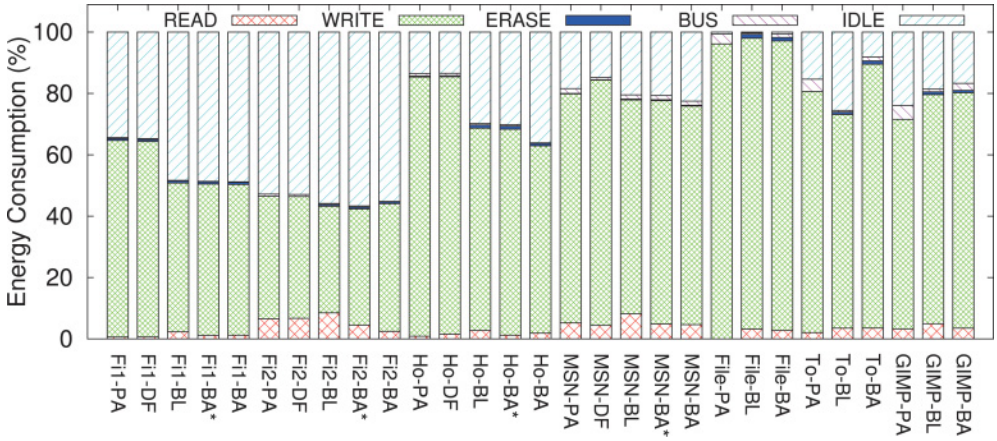


Fig. 10. Details of energy consumption by NAND operation (Fi: Financial, Ho: Homes, MSN: MSNfs, File: FileZilla, To: Torrent, PA: page mapping, DF: DFTL, BL: block mapping, BA: BAST).

the storage device. We have observed that block mapping and two hybrid mappings exhibit significantly longer I/O latencies than page mapping does. Block mapping, BAST*, and BAST consume 9.2%, 7.9%, and 21.9% more energy, respectively, than page mapping does. Sensitivity analysis shows that the accesses to the spare area during the replacement block management in block mapping negatively affect both the performance and the energy consumption; the effect is far graver in energy consumption than in performance.

5.3. Energy Consumption Breakdown

In Figure 10, we show the breakdown of the total energy consumption by the NAND flash operation type. The idle time energy consumption in the figure excludes the energy consumption when an SSD itself is in an idle state; that is, it only includes the aggregated energy consumption of each idle flash chip when at least one flash chip is executing a NAND operation. This is to accurately extract the amount of energy

Table VII. Energy Consumption for Each NAND Operation (%)
(Workload: Financial1, Stripe Size: Page)

	Read	Write	Erase	Idle	BUS
Page	3.3	65.8	0.3	29.8	0.9
DFTL	3.3	66.8	0.3	29.0	0.6
Block	5.4	54.7	0.7	38.6	0.6
BAST*	2.9	56.8	0.7	39.0	0.6
BAST	2.5	55.8	0.6	40.7	0.4

Table VIII. NAND Operation Counts and Energy Consumption (Page Mapping)

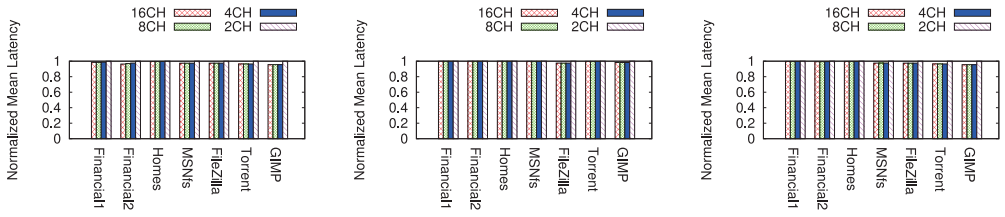
	Page Read ($\times$1,000)	Page Write ($\times$1,000)	Idle Time Energy Cons.	Total Energy Cons.	Ratio (%)
MSN	1,959	1,175	31J	169J	18.6
FileZilla	1	1,296	287mJ	141J	0.2

needed to maintain idle flash chips in an idle state.[5] A common observation throughout all mapping schemes and workloads is that the energy consumption involved in write operations and during idle time accounts for a dominant portion of the total energy consumption. On average, each of them occupies 66% and 29% of the total energy consumption, respectively. Even with the read-intensive workload, Financial2, where 79% of the total I/O operations are read operations, write operations consume about 40% of the total energy, on average, while read operations consume only 7% of the total energy, on average. This is because a read operation consumes only about 1/18 of the energy compared to a write operation. Current consumptions and voltages are the same for read, write, and erase operations. The duration of each operation is different. Read and write operations take 50μsec and 900μsec, respectively. A write operation consumes 18$\times$ more energy than a read operation does. While a single erase operation consumes 2$\times$ more energy (132nJ) than a write operation, the aggregated energy consumed by all erase operations occupies only 0.5% of the total energy consumption due to the very small number of erase operations.

Table VII shows the energy consumed by each NAND operation as a percentage of the total energy consumption. In page mapping and DFTL, write operations consume a relatively larger fraction of energy. In block mapping and two hybrid mappings, flash memories spend a relatively large fraction of energy in an idle state. This is because the two schemes, page mapping and block mapping, have different degrees of parallelism across flash chips. In multichannel/multiway SSDs, if the channels are not fully utilized, some flash devices sit idle while the other flash devices are under operation. We observed that the total energy consumption in the idle flash devices far exceeds the energy consumption of a single page write operation. Therefore, in terms of energy consumption, it is important to minimize the idle flash chips via maximizing parallel operations among flash chips. In page mapping, sequential page writes can be distributed across channels, whereas in block mapping, sequential pages are written in the same block until the block becomes full. Hence, for small sequential writes (less than a block), block mapping cannot fully exploit channel parallelism and places most flash chips in an idle state.

Table VIII shows the number of individual flash operations, idle time energy consumption (A), total energy consumption (B), and ratio of idle time energy consumption to the total energy consumption (A/B). The MSNfs workload issues mainly small and

[5]Note that when an SSD is in an idle state, we can safely turn off the entire flash memory part in the SSD to save energy.



(a) Page-level FTL (b) Block FTL (c) BAST FTL

Fig. 11. Normalized mean response time in each channel/way architecture.

random write requests, while the FileZilla workload issues mainly large and sequential write requests. The ratio of the idle time energy consumption to the total energy consumption in each workload is about 18.6% and 0.2%, respectively. We can see the effect of the idle time on the total energy consumption. MSNfs issues 2.5× more I/Os than FileZilla, most of which are read operations. A single page write operation consumes about 18× more energy than a single page read operation. MSNfs consumes only 19.5% more energy than FileZilla in total, due to the high idle time energy consumption.

Based on the previous observations and analysis, we can identify an important SSD design objective in terms of energy consumption. The FTL algorithm should carefully be designed to increase the degree of parallel operations to minimize the idle time of flash chips. Additionally, the SSD architecture should be designed to allow the flash packages to be turned off when they are in an idle state.

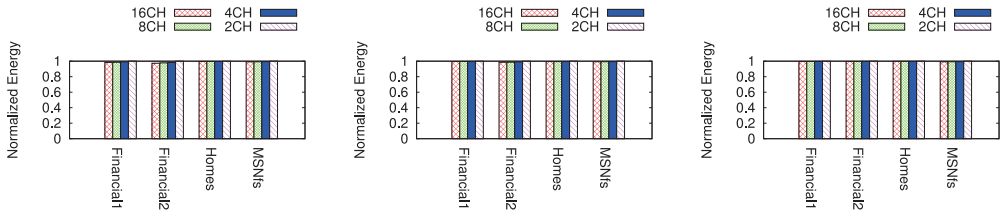
5.4. The Internal Parallelism

To investigate the effect of the channel and way configuration on both the I/O response time and the energy consumption, we measured the response time and energy consumption under various channel/way architectures (from 2×8 to 16×1).[6] We assumed that physical pages are allocated in a channel-major round-robin fashion for write requests.

Figure 11 shows the average response time in each channel/way configuration. The response times are normalized to a 2 × 8 configuration (two channels and eight ways). A common observation among all mapping schemes is that the response times in 4 × 4, 8 × 2, and 16 × 1 configurations are almost the same in most workloads. This is very interesting because the increased channel parallelism brings marginal improvement to the I/O response time, contrary to common perception. We found the reason in the dynamics of both the channel switching delay and way switching delay (the time for the flash controller to transfer data from the SSD buffer to the page register in the flash chip). When the number of channels becomes large enough, the aggregate channel switch delay hides the way switching delay in each channel. In our experiment, the channel switching and way switching delay are 30μsec and 82μsec [Yoo et al. 2011], respectively.

Let us provide an example. We examined detailed latency in issuing a 32KB (eight page × 4KB) write under different channel/way configurations. In a 2 × 8 configuration, four pages are assigned to two channels, in a round-robin fashion. Since the latency to issue two page write commands to two different channels is 60μsec (=2 × 30μsec), the SSD controller needs to wait 22μsec (=82μsec –60μsec) until the first channel is ready to accept a new command. Hence, to issue eight page write commands to two channels, there occur six way switches (6 × 22μsec = 132μsec), in addition to the eight channel

[6]m by n denotes m-channel and n-way configuration.



(a) Page-level FTL (b) Block FTL (c) BAST FTL

Fig. 12. Normalized energy consumption in each channel/way architecture.

switches (=240μsec). However, in a 4 $\times$ 4 configuration, the time to issue four page write commands to four different channels is 120μsec (=4 $\times$ 30μsec), longer than the way switching delay (82μsec). By the time the fourth page is issued to the channel 4, the channel 1 will be ready to accept a new command. In this case, we can issue four page write commands without any delay. The total delay in issuing eight page write commands in a 4 $\times$ 4 configuration will be only 240μsec, shorter than the delay in a 2 $\times$ 8 configuration. When the number of channels is larger than four, the total delay time is the same for 8 $\times$ 2, 16 $\times$ 1, and 4 $\times$ 4 configurations. Therefore, to determine the channel/way configuration in an SSD, we need to properly incorporate the channel and way switching delays to eliminate unnecessary delay caused by switching overhead.

Figure 12 shows the normalized average energy consumption under various channel/way configurations. We normalized the average energy consumption against that of the 2 $\times$ 8 configuration. While the average energy consumption slightly decreases as the number of channels increases, the differences are negligible. The result of the performance experiment includes not only the workloads with small-size I/Os but also the workloads with large-size I/Os. For the FileZilla workload, the average size of write operations is 437Kbyte. The average read size of the GIMP workload is 136Kbyte. The channel/way configuration does not notably affect the energy consumption provided that the number of flash chips is the same.

5.5. Effect of Channel/Way Switch Delay

We measured the performance and energy consumption of four different combinations of the channel/way switch delay: 15/41, 30/82, 30/123, and 30/164μsec. Figure 13 and Figure 14 illustrate the normalized energy consumption and normalized response time under four different combinations of channel and way switch delays, respectively. In these figures, there are seven groups of bars, and there are four bars in each group. Each group denotes the experiment result for each of seven traces. In each group, the results are normalized against the result of the longest channel and way switch configuration (30/164μsec) in the group. As shown in Figure 13, energy consumption is not sensitive to the changes in channel and way switch delay. However, the response time is sensitive to the changes in channel and way switch delay.

5.6. Peak Energy Consumption

We examined the fraction of time the SSD controller consumes current at its maximum level. The length of this period is equivalent to the time during which the maximum number of flash chips is being programmed in parallel. We used four different channel configurations under six workloads. The key technical issue in this experiment is the effect of page write latency. We used four different channel configurations: 16 $\times$ 1, 8 $\times$ 2, 4 $\times$ 4, and 2 $\times$ 8. Existing works do not take into account the delay involved in switching the channels and ways [Park et al. 2009a; Agrawal et al. 2008]. However, it is found

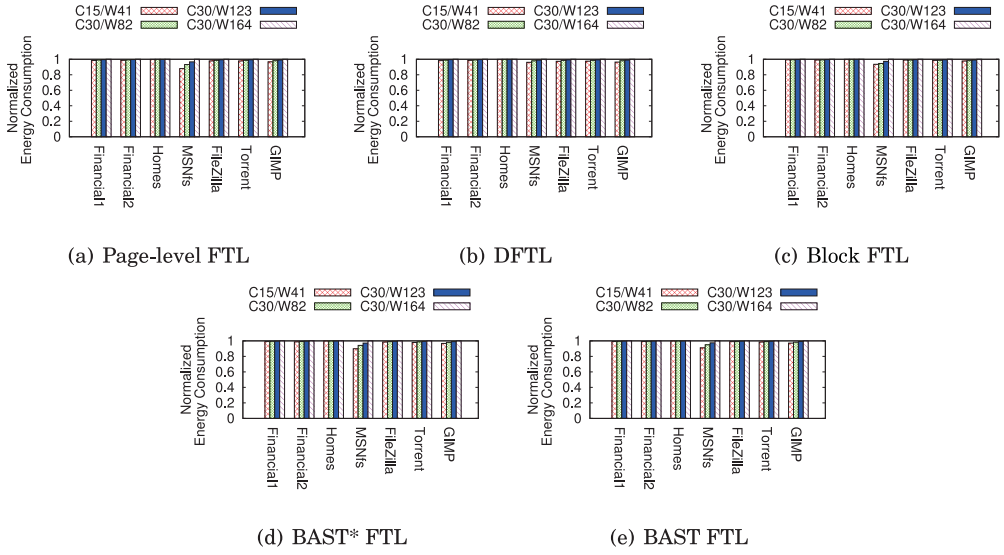


Fig. 13. Normalized energy consumption (C: channel switch delay, W: way switch delay (μsec)).

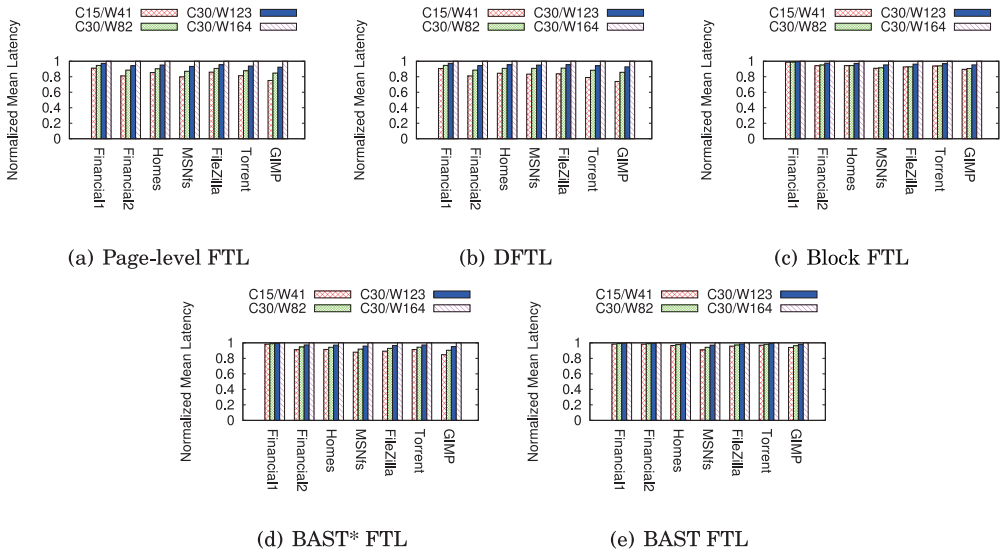
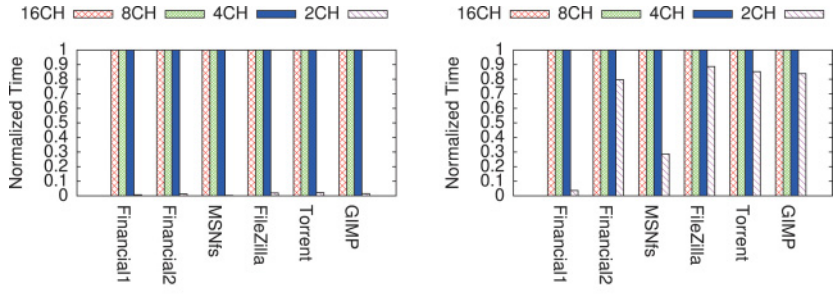


Fig. 14. Normalized mean latency (C: channel switch delay, W: way switch delay (μsec)).

that the channel switch delay cannot be ignored in studying the effect of parallelism [Yoo et al. 2013]. The most *effective* parallelism is governed not only by the number of channels and ways but also by the page write latency. To fully exploit the channel parallelism, the total time to interleave the write requests across the channels should be longer than the delay of switching the NAND die in a channel (way switch delay). Further, to fully exploit the hardware parallelism, the sum of latencies in switching the channels and ways should be longer than the time spent on writing a single page. For example, assume that a 64KByte write command has arrived from the host and the SSD has four channels and four ways. With a 4KByte page size, the write command



(a) Write latency: 500μs (b) Write latency: 900μs

Fig. 15. Max current consumption time in page FTL (max current is 477mA in experiment parameter).

will be split into 16 NAND write requests. Assuming plain page mapping, these NAND operations will be issued to each channel in round-robin fashion. The first and the fifth write operations will be designated to the same channel, and so will the second and the sixth operations. Given a 30msec channel switch delay [Yoo et al. 2011], the first and the fifth write requests will be 120msec apart. If the way switch delay is shorter than 120msec, the first channel will be ready to accept a new write command when the fifth request arrives. Otherwise, the fifth write request has to wait until the NAND controller is ready to write a page to a new block. The first and the 17th page writes will go to the same die in a 4×4 configuration. The 17th write command cannot start until the first write request completes. The internal parallelism should scale with the page write latency and the equivalent page size.

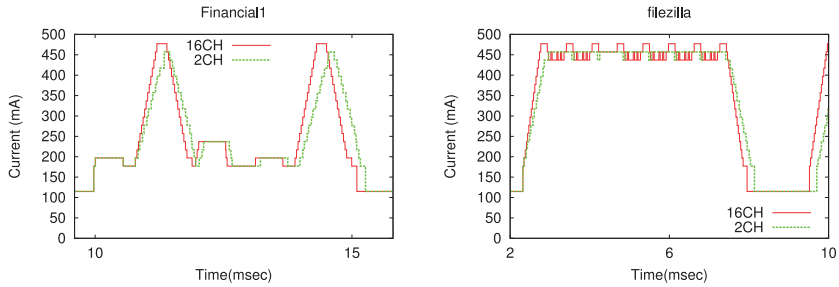
Figure 15 illustrates the fraction of time during which the SSD consumes current at its maximum level. We ran the experiment with two page write latencies, 500μsec and 900μsec. It is normalized to the 16-channel SSD. The channel switch delay and way switch delay are set to 30μsec and 82μsec, respectively [Yoo et al. 2011]. As long as the number of channels is larger than or equal to three, that is, $\lceil \frac{86}{30} \rceil$, we can saturate the channel. Therefore, four-, eight-, and 16-channel SSDs yield the same behavior in both Figure 15(a) and Figure 15(b). With two-channel configuration, each channel is underutilized due to the way switching delay. Therefore, the two-channel SSD spends a smaller fraction of time at the maximum current consumption level than the others. Let us examine the effect of page write latency; with larger page write latency (900μsec), there are more outstanding concurrent write operations. Therefore, SSDs spend a longer time at the maximum current consumption level (Figure 15(b)). On the other hand, in a 2×8 configuration, the SSD rarely consumes the maximum current when the page write latency is 500μsec (Figure 15(a)).

For internal parallelism of an SSD, there exist energy/performance tradeoffs. Increasing the internal parallelism improves the performance, but it also increases the peak energy consumption. Yoo et al. [2011] suggested the notion of *Power Budget,* which is the maximum tolerable peak energy consumption (or current consumption) for an SSD. They propose that there should be an interface to inform the SSD of its Power Budget and that the firmware of an SSD should be designed to dynamically adjust the parallelism degree subject to its Power Budget. Several works have proposed dynamically throttling the transfer rate or parallelism degree of an SSD to regulate the temperature of the SSD [Park et al. 2009b; Lee et al. 2013]. To properly design the SSD internal parallelism, we need to incorporate the channel switch delay, way switch delay, and page write latency.

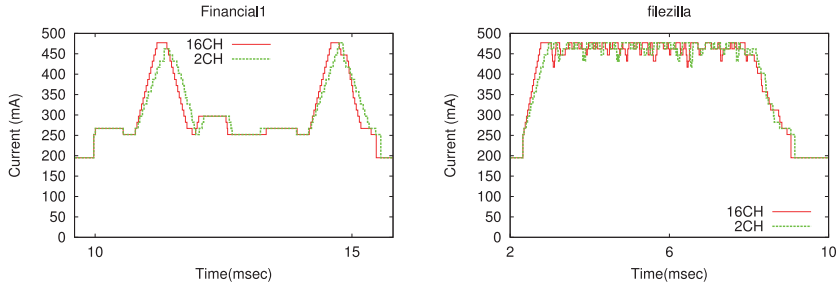
Peak energy consumption is an important factor in designing the SSD controller circuit. We examined the peak energy consumption behavior under different degrees

Table IX. Page Program (Write) Latency

Page Size	Write Latency (μsec)
2KB	200 [Samsung 2005]
4KB	900 [Intel 2009a]
8KB	1300 [Samsung 2012]
16KB	1600 [Micron 2013]



(a) Financial1 (Write latency < Sum of Way Delay) (b) FileZilla (Write latency < Sum of Way Delay)



(c) Financial1 (Write latency > Sum of Way Delay) (d) FileZilla (Write latency > Sum of Way Delay)

Fig. 16. Time series of current consumption with different page trite times: 500μsec versys 900μsec.

of parallelism: two-channel and 16-channel SSDs. We used two workloads: Financial1 and FileZilla. The Finanacial1 workload exhibits small random writes with high temporal locality. Most of the I/O accesses are focused on the small region, and therefore, the SSD cannot fully exploit its internal parallelism. The side benefit of this is that the SSD rarely reaches its peak energy consumption. For the FileZilla workload, current consumption of the SSD stays at its peak for most of the time. This is because the SSD controller fully exploits its internal parallelism to sequentially write the files it is downloading. In terms of aggregate energy consumption, as long as the degree of internal parallelism remains the same, how to distribute the parallelism among the channels and ways hardly matters; that is, a two-channel by eight-way configuration and an eight-channel by two-way configuration yield the same aggregate energy consumption. While SSDs with more channels spend a larger fraction of time at the peak current level, we believe the difference is marginal.

In both workloads, the SSD with large-size pages (longer write latency) consumes more energy. The difference in energy consumption becomes more visible in the random workloads (Figure 16(a) and Figure 16(c)) than in the workloads with large sequential writes (Figure 16(b) and Figure 16(d)). Considering that NAND page size is ever increasing these days and that an NAND device with a 16KByte page size is being

manufactured, we would like to examine the effect of the NAND flash page size on the energy consumption and performance when the filesystem block size is a fraction of the NAND flash page size. Most of the modern filesystems use a 4KByte block size to align it with the page frame size of the main memory. From an energy consumption perspective, it should be mandatory that the SSD controller implements subpage mapping.

Peak energy consumption does not change with respect to the channel/way combination. Peak energy consumption is governed by the maximum number of flash chips that can be programmed in parallel. As long as the number of flash chips that can be programmed in parallel remains the same, how they are arranged in channel/way combination does not affect the peak energy consumption. However, the duration at which the SSD consumes energy at its maximum level varies subject to the channel/way configuration. Also, total energy consumption is governed by the total amount of read and write operations and therefore is not subject to the channel/way configurations of NAND flash memory.

6. RELATED WORK

Software components of an SSD include address mapping, garbage collection, and wear leveling. Numerous works have been proposed in the area of address mapping [Hu et al. 2010; Wei et al. 2011; Liu et al. 2012b; Chen et al. 2011b; Gupta et al. 2009], wear leveling [Chen et al. 2011a; Wu et al. 2011; Murugan and Du 2011; Shmidt 2002], and garbage collection [Liu et al. 2012a; Debnath et al. 2011]. These algorithms can be executed either in the microcontroller of an SSD or in the CPU of the host [Fusion-IO 2011]. A single NAND flash chip is not as fast as an HDD and can exhibit an approximately 40MByte/sec write bandwidth [Strande et al. 2012]. To improve the performance, modern SSDs, without any exception, arrange the NAND flash chip using multiple channels and ways [Park et al. 2009a]. With multichannel and multiway configuration, the SSD controller can perform multiple NAND operations concurrently, which results in higher performance. Pages across the multiple channels can negatively affect the overall SSD performance [Yoo et al. 2013]. There have been a number of works on building an energy-efficient system with SSDs [Strande et al. 2012]. The energy consumption aspect of HDDs has been under intense research for the last two decades and has now reached sufficient maturity. These works propose to stop the spindle and/or to shut down the circuitry of the hard disk controller. It has also been proposed to reduce the rotational speed to save energy [Yada et al. 2000]. Bucy et al. [2009] developed a DiskSim-based simulator to estimate the energy consumption of HDDs for a given workload. Jung et al. [2012] developed NANDFlashSim for NAND flash devices.

With SSD-based storage devices, a system can become more energy efficient than with an HDD in terms of performance [Schall et al. 2010; Seo et al. 2008], that is, higher IOPS/J, but it becomes denser in terms of energy consumption. Energy density of an SSD, that is, J/in^3, is orders of magnitude larger than that of an HDD, and the peak energy consumption of an SSD is much higher than that of an HDD [Inoue et al. 2011], which raises the heat dissipation issue. A number of works are dedicated to regulating the peak energy consumption via throttling the data transfer rate or limiting the number of flash memory chips programmed in parallel [Park et al. 2009b; Lee et al. 2013]. Compared to HDDs, SSDs will make CPU spend less time waiting for an I/O. This may escalate the peak energy consumption of a given system. Special care needs to be taken to regulate the peak energy consumption of the entire system.

7. CONCLUSION

Researchers as well as practitioners do their best to get better performance out of SSDs. This is achieved via increasing internal parallelism, adopting large-size buffers,

increasing the areal density of the NAND flash cells, and so forth. As a result, modern SSDs exhibit spectacular performance compared to the legacy HDDs. SSDs are more energy efficient than HDDs from a byte/J point of view. However, the aggregate and peak energy consumptions of an SSD become much denser than those of an HDD (measured in J/sec and J/cm^3, respectively). We believe that the energy aspect of an SSD design deserves more interest from the community. This work focuses on the energy consumption behavior of an SSD under various design choices. We varied the latency of NAND operations, the number of channels, and FTL algorithms and examined the detailed energy consumption behavior of an SSD. We found that DFTL is the most energy-efficient scheme due to its good write amplification behavior and the small memory footprint. We found that aggregate energy consumption is not sensitive to the channel/way configuration but is rather governed by the write volume. While a single erase operation consumes more energy than a single write operation, the aggregate energy consumption of all erase operations constitutes less than 1% of the total energy consumed by NAND flash. Peak energy consumption is governed by internal parallelism. When it is required to increase the internal parallelism, it is better to increase way parallelism than channel parallelism from a peak current consumption point of view. The result of this work provides an important guideline in designing energy-efficient SSDs.

REFERENCES

Nitin Agrawal, Vijayan Prabhakaran, Ted Wobber, John D. Davis, Mark Manasse, and Rina Panigrahy. 2008. Design tradeoffs for SSD performance. In *Proceedings of the USENIX 2008 Annual Technical Conference on Annual Technical Conference*. USENIX, 57–70.

Amir Ban. 1995. Flash file system. *U.S. Patent 5 404 485* (April 4, 1995).

Amir Ban. 1999. Flash file system optimized for page-mode flash technologies. *U.S. Patent 5 937 425* (Aug. 10, 1999).

John S. Bucy, Gregory R. Ganger, and et al. 2009. The DiskSim Simulation Version 4.0. Available at http://www.pdl.cmu.edu/DiskSim.

Feng Chen, Tian Luo, and Xiaodong Zhang. 2011a. CAFTL: A content-aware flash translation layer enhancing the lifespan of flash memory based solid state drives. In *Proceedings of the 9th USENIX Conference on File and Storage Technologies (FAST'11)*. USENIX Association, 6–6.

Zhiguang Chen, Nong Xiao, Fang Liu, and Yimo Du. 2011b. PBFTL: The page to block mapping FTL with low response time. In *Proceedings of the 2011 IEEE 19th International Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS'11)*. Singapore, 475–477. DOI:http://dx.doi.org/10.1109/MASCOTS.2011.31

Biplob Debnath, Srinibasan Krishnan, Weijun Xiao, David J. Lilja, and David H. C. Du. 2011. Sampling-based garbage collection metadata management scheme for flash-based storage. In *Proceedings of the 2011 IEEE 27th Symposium on Mass Storage Systems and Technologies (MSST'11)*. 1–6. DOI:http://dx.doi.org/10.1109/MSST.2011.5937228

ESOSLab. 2012. Linux Application Trace. Retrieved from http://esos.hanyang.ac.kr/sub/ssd_trace_data.zip.

Eitan Frachtenberg, Ali Heydari, Harry Li, Amir Michael, Jacob Na, Avery Nisbet, and Pierluigi Sarti. 2011. High-efficiency server design. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, New York, NY, 27.

Fusion-IO. 2011. The fusion-io difference. In *White Paper, Fusion-IO (WPCH031511)*.

Eran Gal and Sivan Toledo. 2005. Algorithms and data structures for flash memories. *ACM Computer Surveys* 37, 2 (June 2005), 138–163. DOI:http://dx.doi.org/10.1145/1089733.1089735

Gary Grider. 2011. ExaScale FSIO - Can we get there? Can we afford to? Retrieved from http://snapi2011.cis.fiu.edu/uploads/General/1.Grider.pdf.

Laura M. Grupp, Adrian M. Caulfield, Joel Coburn, Steven Swanson, Eitan Yaakobi, Paul H. Siegel, and Jack K. Wolf. 2009. Characterizing flash memory: Anomalies, observations, and applications. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 42)*. ACM, New York, NY, 24–33. DOI:http://dx.doi.org/10.1145/1669112.1669118

Aayush Gupta, Youngjae Kim, and Bhuvan Urgaonkar. 2009. DFTL: A flash translation layer employing demand-based selective caching of page-level address mappings. In *Proceeding of the 14th International*

Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'09). ACM, Washington, DC, 229–240. DOI:http://dx.doi.org/10.1145/1508244.1508271

Jiahua He, Arun Jagatheesan, Sandeep Gupta, Jeffrey Bennett, and Allan Snavely. 2010. DASH: A recipe for a flash-based data intensive supercomputer. In *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC'10)*. IEEE Computer Society, Washington, DC, 1–11. DOI:http://dx.doi.org/10.1109/SC.2010.16

Yang Hu, Hong Jiang, Dan Feng, Lei Tian, Shuping Zhang, Jingning Liu, Wei Tong, Yi Qin, and Liuzheng Wang. 2010. Achieving page-mapping FTL performance at block-mapping FTL cost by hiding address translation. In *Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST'10)*. 1–12. DOI:http://dx.doi.org/10.1109/MSST.2010.5496970

IC-insights. December 19, 2012. Total flash memory market will surpass DRAM for first time in 2012. In *Research Bulletin, IC insights*.

Takuro Inoue, Makoto Ikeda, Tomoya Enokido, Ailixier Aikebaier, and Makoto Takizawa. 2011. A power consumption model for storage-based applications. In *Proceedings of the 2011 International Conference on Complex, Intelligent and Software Intensive Systems (CISIS'11)*. IEEE, 612–617.

Intel. 2012. Intel Solid-State Drive 320 Series and Dell PowerEdge. Retrieved from http://www.intel.com.br/content/dam/www/public/us/en/documents/technology-briefs/ssd-320-dell-poweredge-brief.pdf.

Intel. 2009a. MD332B NAND Flash Memory. Preliminary Datasheet.

Intel. 2009b. Intel X25-M and X18-M Mainstream SATA Solid-State Drives. Retrieved from http://www.intel.com/design/flash/nand/mainstream/technicaldocuments.htm.

Kim Jesung, Kim Jong Min, S. H. Noh, Min Sang Lyul, and Cho Yookun. 2002. A space-efficient flash translation layer for CompactFlash systems. *IEEE Transactions on Consumer Electronics* 48, 2 (2002), 366–375.

Myoungsoo Jung, Ellis Herbert Wilson, David Donofrio, John Shalf, and Mahmut T. Kandemir. 2012. NANDFlashSim: Intrinsic latency variation aware NAND flash memory system modeling and simulation at microarchitecture level. In *Proceedings of the 2012 IEEE 28th Symposium on Mass Storage Systems and Technologies (MSST'12)*. IEEE, 1–12.

Swaroop Kavalanekar, Bruce Worthington, Qi Zhang, and Vishal Sharda. 2008. Characterization of storage workload traces from production Windows Servers. In *Proceedings of the 2008 IEEE International Symposium on Proceedings of Workload Characterization (IISWC 2008)*. 119–128. DOI:http://dx.doi.org/10.1109/IISWC.2008.4636097

Ricardo Koller and Raju Rangaswami. 2010. I/O deduplication: Utilizing content similarity to improve I/O performance. *Transactions on Storage* 6, 3, Article 13 (Sept. 2010), 26 pages. DOI:http://dx.doi.org/10.1145/1837915.1837921

Tei-Wei Kuo, Yuan-Hao Chang, Po-Chun Huang, and Che-Wei Chang. 2008. Special issues in flash. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design, 2008 (ICCAD'08)*. San Jose, CA, USA, 821–826. DOI:http://dx.doi.org/10.1109/ICCAD.2008.4694174

Sungjin Lee, Taejin Kim, Ji-Sung Park, and Jihong Kim. 2013. An integrated approach for managing the lifetime of flash-based SSDs. In *Proceedings of the Conference on Design, Automation and Test in Europe*. EDA Consortium, 1522–1525.

Duo Liu, Yi Wang, Zhiwei Qin, Zili Shao, and Yong Guan. 2012a. A space reuse strategy for flash translation layers in SLC NAND flash memory storage systems. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 20, 6 (June 2012), 1094–1107. DOI:http://dx.doi.org/10.1109/TVLSI.2011.2142015

Duo Liu, Tianzheng Wang, Yi Wang, Zhiwei Qin, and Zili Shao. 2012b. A block-level flash memory management scheme for reducing write activities in PCM-based embedded systems. In *Proceedings of the Design, Automation Test in Europe Conference Exhibition (DATE'12)*. 1447–1450.

Micron. 2013. 128Gb to 1Tb Asynchronous/Synchronous NAND, datasheet, Rev. G. 2013.

Muthukumar Murugan and David H. C. Du. 2011. Rejuvenator: A static wear leveling algorithm for NAND flash memory with minimized overhead. In *Proceedings of the 2011 IEEE 27th Symposium on Mass Storage Systems and Technologies (MSST'11)*. 1–12. DOI:http://dx.doi.org/10.1109/MSST.2011.5937225

Dushyanth Narayanan, Eno Thereska, Austin Donnelly, Sameh Elnikety, and Antony Rowstron. 2009. Migrating server storage to SSDs: Analysis of tradeoffs. In *Proceedings of the 4th ACM European Conference on Computer Systems*. ACM, 145–158.

Xiang Ni, Esteban Meneses, and Laxmikant V. Kale. 2012. Hiding checkpoint overhead in HPC applications with a semi-blocking algorithm. In *Proceedings of the 2012 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 364–372.

ONFI. 2011. Open NAND Flash Interface (ONFI) Specification 3.0. Retrieved from http://www.onfi.org/specifications.

Sang-Hoon Park, Seung-Hwan Ha, Kwanhu Bang, and Eui-Young Chung. 2009a. Design and analysis of flash translation layers for multi-channel NAND flash-based storage devices. *IEEE Transactions on Consumer Electronics* 55, 3 (2009), 1392–1400.

Jinha Park, Sungjoo Yoo, Sunggu Lee, and Chanik Park. 2009b. Power modeling of solid state disk for dynamic power management policy design in embedded systems. In *Software Technologies for Embedded and Ubiquitous Systems*. Vol. 5860. Springer, 24–35. DOI:http://dx.doi.org/10.1007/978-3-642-10265-3_3

Padmanabhan Pillai, Michael Kaminsky, Michael A. Kozuch, and David G. Andersen. 2012. FAWNSort: Energy efficient Sorting of 10GB, 100GB, and 1TB. Winner of 2012 10GB, 100GB, and 1TB, Joulesort Daytona and Indy categories. Retrieved from http://softbenchmark.org/fawnsort-joulesort-2012.pdf.

Meikel Poess and Raghunath Othayoth Nambiar. 2010. Tuning servers, storage and database for energy efficient data warehouses. In *Proceedings of the 2010 IEEE 26th International Conference on Data Engineering (ICDE'10)*. IEEE, 1006–1017.

Samsung. 2012. 64Gb A-die Toggle NAND Flash, datasheet Rev. 1.1. 2012.

Samsung. 2005. K9XXG08UXM NAND Flash Memory, Preliminary Datasheet.

Daniel Schall, Volker Hudlet, and Theo Harder. 2010. Enhancing energy efficiency of database applications using SSDs. In *Proceedings of the 3rd C* Conference on Computer Science and Software Engineering (C3S2E'10)*. ACM, New York, NY, 1–9. DOI:http://dx.doi.org/10.1145/1822327.1822328

Euiseong Seo, Seon-Yeong Park, and Bhuvan Urgaonkar. 2008. Empirical analysis on energy efficiency of flash-based SSDs. In *Proceedings of the 2008 Conference on Power Aware Computing and Systems (HotPower'08)*. USENIX Association, San Diego, California, 17–17.

Hyotaek Shim, Bon-Keun Seo, Jin-Soo Kim, and Seungryoul Maeng. 2010. An adaptive partitioning scheme for DRAM-based cache in Solid State Drives. In *Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*. 1–12. DOI:http://dx.doi.org/10.1109/MSST.2010.5496995

Dmitry Shmidt. 2002. TrueFFS wear-leveling mechanism. In *Technical Report, M-systems (TN-DOC-017)*.

SPC. 2009. UMASS TRACE REPOSITORY. Retrieved from http://traces.cs.umass.edu/.

Shawn M. Strande, Pietro Cicotti, Robert S. Sinkovits, William S. Young, Rick Wagner, Mahidhar Tatineni, Eva Hocks, Allan Snavely, and Mike Norman. 2012. Gordon: Design, performance, and experiences deploying and supporting a data intensive supercomputer. In *Proceedings of the 1st Conference of the Extreme Science and Engineering Discovery Environment: Bridging from the eXtreme to the Campus and Beyond (XSEDE'12)*. ACM, Article 3, 8 pages. DOI:http://dx.doi.org/10.1145/2335755.2335789

Dimitris Tsirogiannis, Stavros Harizopoulos, and Mehul A. Shah. 2010. Analyzing the energy efficiency of a database server. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*. ACM, 231–242.

Qingsong Wei, Bozhao Gong, S. Pathak, B. Veeravalli, LingFang Zeng, and K. Okada. 2011. WAFTL: A workload adaptive flash translation layer with data partition. In *2011 IEEE 27th Symposium on Mass Storage Systems and Technologies (MSST)*. Denver, CO, USA, 1–12. DOI:http://dx.doi.org/10.1109/MSST.2011.5937217

Qi Wu, Guiqiang Dong, and Tong Zhang. 2011. Exploiting heat-accelerated flash memory wear-out recovery to enable self-healing SSDs. In *Proceedings of the 3rd USENIX Conference on Hot Topics in Storage and File Systems (HotStorage 2011)*. USENIX, 4–4. http://dl.acm.org/citation.cfm?id=2002218.2002222

H. Yada, H. Ishioka, T. Yamakoshi, Y. Onuki, Y. Shimano, M. Uchida, H. Kanno, and N. Hayashi. 2000. Head positioning servo and data channel for HDDs with multiple spindle speeds. *IEEE Transactions on Magnetics* 36, 5 (2000), 2213–2215.

Balgeun Yoo, Youjip Won, Seokhei Cho, Sooyong Kang, Jongmoo Choi, and Sungroh Yoon. 2011. SSD characterization: From energy consumption's perspective. In *Proceedings of USENIX HotStorage*. USENIX.

Jinsoo Yoo, Youjip Won, Joongwoo Hwang, Sooyong Kang, Jongmoo Choi, Sungroh Yoon, and Jaehyuk Cha. 2013. VVSIM: Virtual machine based SSD simulator. In *Proceedings of the 2013 IEEE 29th Symposium on Proceeding of Mass Storage Systems and Technologies (MSST'13)*. IEEE.

Received July 2013; revised April 2014; accepted July 2014