IEICE TRANS. INF. & SYST., VOL.E96–D, NO.5 MAY 2013

PAPER

Power Failure Protection Scheme for Reliable High-Performance Solid State Disks

Kwanhu BANG[†a)], **Kyung-Il IM**[††], **Dong-gun KIM**[†††], **Sang-Hoon PARK**[†], ***Nonmembers***, ***and*** **Eui-Young CHUNG**[†b)], ***Member***

SUMMARY Solid-state disks (SSDs) have received much attention as replacements for hard disk drives (HDDs). One of their noticeable advantages is their high-speed read/write operation. To achieve good performance, SSDs have an internal memory hierarchy which includes several volatile memories, such as DRAMs and SRAMs. Furthermore, many SSDs adopt aggressive memory management schemes under the assumption of stable power supply. Unfortunately, the data stored in the volatile memories are lost when the power supplied to SSDs is abruptly shut off. Such power failure is often observed in portable devices. For this reason, it is critical to provide a power failure protection scheme for reliable SSDs. In this work, we propose a power-failure protection scheme for SSDs to increase their reliability. The contribution of our work is three-fold. First, we design a power failure protection circuit which incorporates super-capacitors as well as rechargeable batteries. Second, we provide a method to determine the capacity of backup power sources. Third, we propose a data backup procedure when the power failure occurs. We implemented our method on a real board and applied it to a notebook PC with a contemporary SSD. The board measurement and simulation results prove that our method is robust in cases of sudden power failure.

key words: *solid state disk, power failure protection, data backup, super-capacitor, rechargeable battery*

1. Introduction

The outstanding features of solid state disks (SSDs) over conventional storage devices – hard disk drives (HDDs) – are high performance, light weight, shock resistance, and low power consumption. These advantages are mainly due to the replacement of mechanical media with electronic storage devices, i.e. NAND flash memories (NFMs). Such features allow SSDs to be widely adopted in various portable devices, such as MP3 players, portable multimedia players, mobile phones, and notebook computers, as well as server systems [1]. On the other hand, disadvantages of NFMs include out-of-place update, a larger erase unit than that of read/write and limited erase cycle. These are factors limiting the further widespread use of SSDs. To compensate for their disadvantages, SSDs adopt an intermediate software layer called the flash translation layer (FTL). Various FTL algorithms have been proposed, and their effectiveness is appreciated in a variety of flash-based storage systems [2]–[4].

In addition, some architectural approaches have been proposed to improve SSDs. One approach is to use volatile memories (DRAMs and SRAMs) as buffers to temporarily store metadata as well as user data. This approach is further enhanced by adding a buffer replacement algorithm (BRA) to the FTL. For instance, some BRAs attempt to improve performance by caching user data in DRAMs. More specifically, these BRA techniques aim to increase the cache hit rate for reducing the number of accesses to NFMs, which are several orders of magnitude slower than DRAMs. The reduction in NFM accesses improves not only performance, but also lifetime of NFMs. BPLRU, CFLRU, and FAB are well-known BRAs, so interested readers are referred to [5]–[7]. One of the most important assumptions behind BRAs is that the power supplied to SSDs is stable. Without this assumption, the updated data in volatile memories will be lost when a sudden power failure occurs.

Unfortunately, there has been little published work on power-failure protection schemes for SSDs. Only a few articles discuss this issue. They report that some commercial SSDs adopt power-failure protection schemes based on super-capacitors. Super-capacitors are only one possible design in power-failure protection schemes and do not address other key features, such as the stability of the power-failure protection scheme or the capacity of the backup power source. The capacity should be large enough to back up the data residing in volatile memories but small enough not to increase the cost very much. Besides, many previous works ignored the power-failure notification scheme and data recovery procedure, which are crucial to the reliability of SSDs having volatile buffers.

In this paper, we propose a power-failure protection scheme for SSDs which covers all the aforementioned issues. For simplicity, we call our method the power failure detection and notification (PFDN) scheme. We propose two different PFDN schemes in this paper. One is based on super-capacitors, similar to previous work, and the other is based on a rechargeable battery. We implemented both schemes on real boards and tested them on a notebook computer. The measurements on these boards and some simulation results support the effectiveness of our method.

The rest of this paper is organized as follows. In Sects. 2 and 3, we provide some preliminaries of SSDs and the works related to a power-failure protection scheme for

Manuscript received July 28, 2011.
Manuscript revised July 26, 2012.

[†]The authors are with the School of Electrical and Electronic Engineering, Yonsei University, Seoul 120–749, Korea.
[††]The author is with the IT Solution Division, Samsung Electronics, Kyunggi-do 443–742, Korea.
[†††]The author is with the Advanced Design System Architecture Team, SK Hynix Semiconductor Inc., Gyeonggi-Do, Korea.
a) E-mail: khbang@dtl.yonsei.ac.kr
b) E-mail: eychung@yonsei.ac.kr

DOI: 10.1587/transinf.E96.D.1078

SSDs, respectively. In Sect. 4, we address the proposed PFDN schemes while focusing on circuit design, backup power capacity, and the power-failure notification and data recovery scheme. In Sect. 5, we discuss the results of subjecting our method to simulations and measurements, followed by a conclusion in Sect. 6.

2. Preliminaries

2.1 SSD Architecture

In this work, we assume a typical SSD architecture, as shown in Fig. 1. However, our method is applicable to any other SSD architectures which have internal volatile memories without loss of generality. In this architecture, SSD consists of three parts – an SSD controller, a cache buffer, and NFMs. Among these, the cache buffer is usually implemented in volatile memories (DRAMs) and the SSD controller includes SRAMs to temporarily store metadata such as address mapping tables. For this reason, these two parts need backup power sources to avoid losing data in volatile memories when power failure occurs. More details of SSD architecture are discussed in [8].

2.2 Metadata in SRAM

An FTL translates a logical address from the host machine to a physical address based on a table called an address mapping table. Whenever a host machine makes a request to an SSD, its FTL always refers to a mapping table for address translation. Hence, it should reside in fast memory such as SRAM. However, on-chip SRAM cannot be large enough to store the whole mapping table in large capacity SSDs. In this case, a part of the table is loaded into SRAM whenever it is necessary and the remaining part resides in the cache buffer, which is much larger than the SRAM. Including the mapping table, an FTL manages many kinds of metadata such as a list of free blocks, the type of a certain block defined by the FTL (log block, data block, or free block). When metadata is lost, there is no way for SSD to respond to the host machine, since mapping information and/or other critical information is missing. For this reason, SSDs are designed to store these data in NFM when they are normally turned off. However, sudden power failure does not allow this data to be safely saved into NFMs.

2.3 Data in the Cache Buffer

There are two types of data stored in the cache buffer. The first is metadata as mentioned in Sect. 2.2. The remaining data type is user data. More exactly, the user data in the cache buffer is a copy of real data which are stored in NFMs.

The efficiency of the cache buffer is closely related to the cache replacement policy, normally referred to as the BRA. They improve SSD performance by increasing cache hit rate with only a marginal increase in FTL overhead. The common assumptions of these BRAs are i) the cache buffer adopts a write-back policy, ii) power failure will not occur. The reason that BRAs make these two assumptions is to maximize SSD performance. However, these assumptions are not valid from a reliability perspective, since a write-back cache buffer which may include new or updated data will lose all its data when power fails. Furthermore, the cache buffer can contain metadata as mentioned in Sect. 2.2. In this case, power failure will cause a fatal error.

3. Related Work

As mentioned in Sect. 2, a backup power source is very important for SSD reliability since SSDs have volatile internal memories. Even though some commercial SSDs already have a super-capacitor based power failure protection scheme, there is little published work on the topic. In this section, we briefly summarize a power failure protection scheme presented in [9] with some limitations of this work.

Figure 2 shows an example of a backup power solution with a super-capacitor [9]. As shown in Fig. 2, the super-capacitor is placed inside of a SSD. The power source (5 V in Fig. 2) is supplied to an SSD which charges the super-capacitor. R_{LIM} is a resistor to prevent in-rush currents. Two diodes isolate two power sources – the main power and super-capacitor. When the main power is supplied, the upper diode D1 is turned on, while the lower diode D2 is turned off. Hence the main power is supplied to the SSD. If the main power fails, the upper diode is turned off and the lower

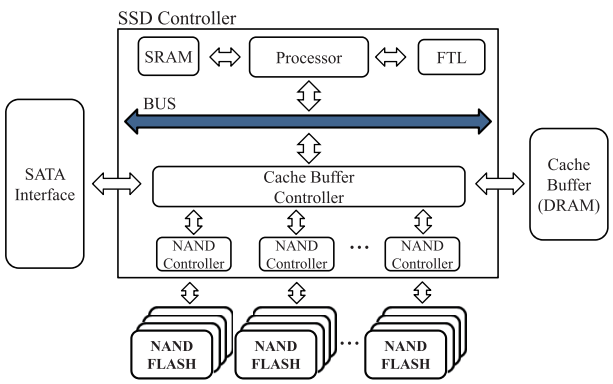


Fig. 1 A typical architecture of NAND flash-based solide state disk [8].

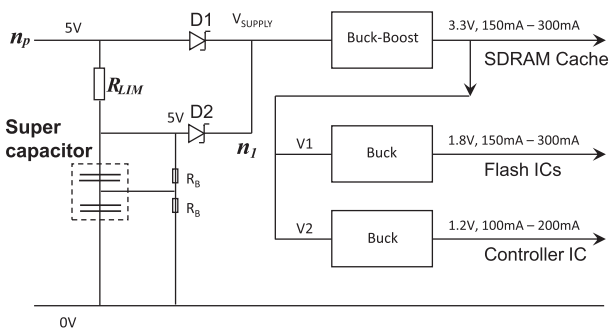


Fig. 2 A super-capacitor based backup power scheme for SSD [9].

diode is turned on. In this case, the super-capacitor becomes the power source.

One major drawback of this scheme is that it cannot notify the change, such as sudden power failure and normal shut down in power sources to the SSD when a system is turned off. Without the notification, the SSD cannot recognize the failure and will fail to back up the data residing on volatile memory while the backup power source is supplying power. Furthermore, when system power failure occurs, current is drained from the super-capacitor. It will flow into the system power line (n_p) as well as the node n_1. Hence, the node n_1 cannot provide sufficient voltage and current to the SSD due to the current drained to n_p. One possible solution is to replace R_{LIM} with a diode to prevent the current from flowing into n_p. Even in this case, the power-failure notification issue is not resolved.

4. Proposed Schemes

4.1 Overview

The proposed power failure protection scheme, PFDN, was implemented and applied to an Intel CPU based notebook computer which adopts an SSD as its storage device. Figure 3 shows a power control circuit of typical notebook computers with the proposed PFDN. The dotted box indicates PFDN, which consists of a control unit and a backup power unit. In normal operation, the power is supplied to the SSD from the wall plug or battery through a DC/DC converter and FETs. The power control system includes a microcontroller which is in charge of a power-up and a power-off sequence.

We propose two different PFDN schemes in this section. One is PFDN with super-capacitors (Sect. 4.2), and the other is PFDN with a rechargeable battery (Sect. 4.3). The reason we propose a rechargeable battery-based PFDN is that some high-end notebook computers already include extra rechargeable batteries, called bridge batteries, for system stability. In this case, we can use the extra battery for the backup power unit of the SSD.

In Sect. 4.4, we address how to determine the capacity of the backup power source from the energy perspective, as well as the peak power perspective. In Sect. 4.5, we present a data backup procedure when power failure occurs.

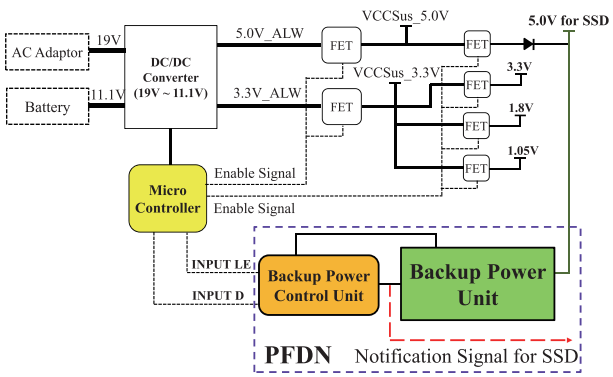


Fig. 3 Power control circuit of typical notebook computer with the proposed PFDN.

4.2 Super-Capacitor Based PFDN

Figure 4 shows a super-capacitor based PFDN which consists of a D-type latch (U1), super-capacitor and three diodes.

In Fig. 4, *INPUT_LE* and *INPUT_D* are from the microcontroller and they are set to 'HIGH' when the system power is normally supplied. In a normal situation, the power to the D-type latch is also supplied from the system power as an SSD, since D2 blocks the power from the super-capacitor to the D-type latch. When the system power fails, D2 is turned on and the microcontroller changes the states of both *INPUT_LE* and *INPUT_D* to 'LOW', hence the output of the D-type latch also becomes 'LOW'. Notice that the state transition of *INPUT_LE* is delayed by a capacitor (C1) in order to satisfy the setup time constraint of the D-type latch. The output of the D-type latch also becomes 'LOW'. This signal is used as a power-failure notification signal and is fed to the SSD through a reserved pin. Inside SSD, this signal triggers the nFIQ (Fast Interrupt reQuest) pin of the SSD controller. nFIQ is active low and initiates the interrupt service routine for copying data from volatile memories to NFMs. While the interrupt service routine is executed, the super-capacitor supplies power to the SSD.

The details of the timing among the signals are shown in Fig. 5. When the system is powered on, the microcontroller waits for a certain amount of time (typically 10 ms) to assert 'HIGH' to *INPUT_LE* and *INPUT_D*, since the

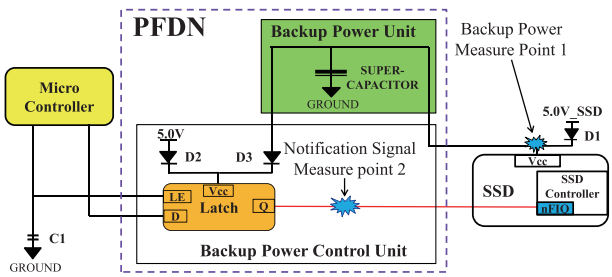


Fig. 4 Super-capacitor based PFDN structure.

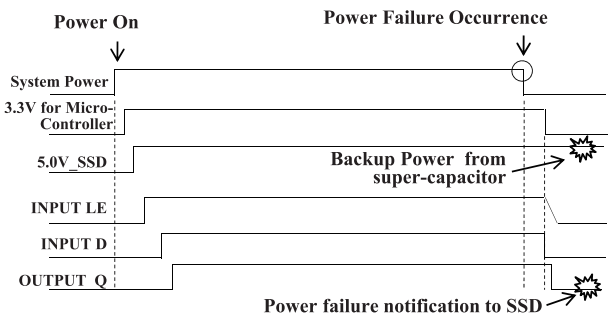


Fig. 5 Timing diagram of super-capacitor based PFDN.

stability of the system power is not guaranteed during the power-up. When the system power fails, both *INPUT_LE* and *INPUT_D* becomes low, since the microcontroller, their driver, is also in a power-off state. As a result, the output of the D-type latch (power-failure notification signal) also becomes 'LOW'. Note that both *INPUT_LE* and *INPUT_D* do not go to 'LOW' when the system is normally turned off. The microcontroller is programmed to set both signals to 'HIGH' for the normal power-off.

4.3 Rechargeable Battery Based PFDN

Another PFDN, a rechargeable battery based PFDN, is shown in Fig. 6. This circuit is similar to the super-capacitor based PFDN, but it is complicated by adopting three FETs. Unlike the super-capacitor, a rechargeable battery continuously drains current as long as there are remaining charges in the battery. To protect from unnecessary current drain during normal operation, we adopt one p-type FET (Q1) and one n-type FET (Q2). Both FETs are gated by the output of the D-type latch. Note that the polarity of *INPUT_D* is reversed for simple implementation. We keep the same polarity for the power-failure notification by selecting the source of Q3 (n-type FET) rather than the output of the D-type latch, hence the SSD is not affected by the polarity issue.

The timing relation of internal signals is shown in Fig. 7. The first six signals behave as they did in Fig. 5, except that *INPUT_D* and *OUTPUT_Q* are inverted. While *OUTPUT_Q* stays 'LOW' (the system power is on), both Q1 and Q2 are turned off. Q3 is also turned off. Therefore the source of Q3, which is the power-failure notification signal, stays 'HIGH' and the interrupt will not occur in this situation. When the system fails, *INPUT_D* becomes 'HIGH', since it is connected to the rechargeable battery through the pull-up resistor R3. *INPUT_LE* will change its state later than *INPUT_D* due to the capacitor C1. Hence, *OUTPUTQ* becomes 'HIGH' and all the FETs are turned on. SSD will be powered by the rechargeable battery through Q1. Also, *OUTPUT_Q* in 'HIGH' turns on Q3 and the source of Q3 becomes 'LOW' due to the large voltage drop across *R*4. At this moment, the source of Q3 notifies the SSD of power failure.

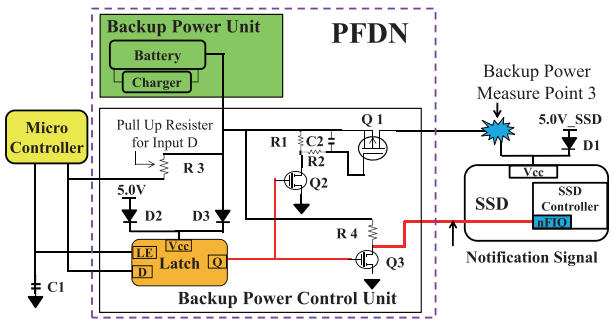


Fig. 6 Rechargeable battery based PFDN structure.

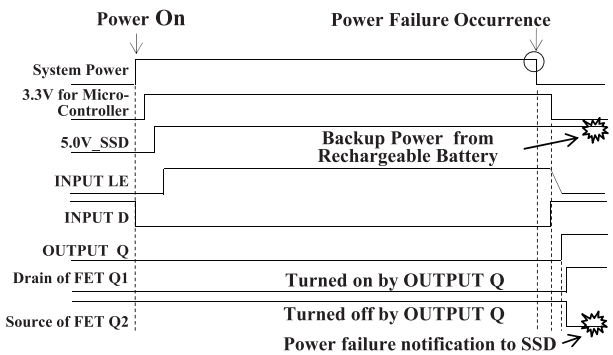


Fig. 7 Timing diagram of rechargeable battery based PFDN.

4.4 Determination of Backup Power Source Capacity

The method discussed in this section is commonly applied to both super-capacitor and rechargeable battery based PFDN.

The capacity of a backup power source should meet two requirements. One is the energy capacity, meaning that the backup power source is large enough to provide appropriate energy for completely moving the updated data in the volatile memories to NFMs. the second factor is peak power capacity for stably supplying power even when the SSD maximally drains current.

We address the energy capacity first in this section. The energy is the product of time and power, hence we need to figure out the total time for transferring the data in volatile memories to NFMs as well as the average power for this operation. Figure 8 depicts the multi-channel/multi-way data transfer timing diagram and the corresponding power profile. A data transfer operation for a single NFM consists of three phases – Cmd, Trans, and Program. These operations are performed in an interleaved manner when there are multiple ways in a single channel. On the other hand, multiple data transfers are performed concurrently when there are multiple channels. Therefore, we need to consider both

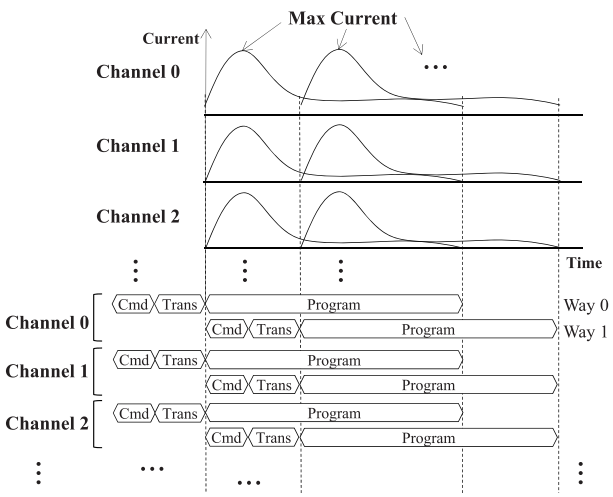


Fig. 8 Data transfers in multi-channel/multi-way SSD with the corresponding power profile [10], [11].

interleaving and concurrency in computing the total data transfer time from the cache buffer to NFMs.

Even though we only need to transfer the updated data in the cache buffer to NFMs, it is hard to identify the updated data in a limited time assuming that the backup power capacity is not large. For this reason, we consider the worst case – all the data in the cache buffer are updated. Fortunately, the work in [12] proposed an analytical model for the data transfer time. More specifically, it models the time required to write a certain amount of data into NFMs in a multi-channel/multi-way architecture, which is given in (1).

$$T_{DE} = \lceil \alpha \rceil \cdot T_{CMD} + \alpha \cdot T_{TRANS} + T_{PROG} + K \cdot (\lceil \alpha / W \rceil - 1) \quad (1)$$

where α is $Q_t/(N_c \cdot S_p)$ and K is $\max\{T_{PROG} - (W-1) \cdot (T_{CMD} + T_{TRANS})\}, 0)$. Q_t is the total amount of data to be written in bytes, N_c is the number of channels, S_p is the size of a page in bytes, and W is the number of ways in each channel. In other words, the first term in (1) means that data are divided by S_p and N_c, and the divided data are concurrently written to NFMs. The second term considers the case where the total data are not a multiple of S_p.

We can compute T_{DE} in (1) by setting Q_t to the given cache buffer size for the worst case, which occurs when the cache buffer is fully filled with dirty pages. If we know the peak ratio of the dirty pages over the cache buffer size, we can scale down Q_t by a scaling factor D so that only dirty pages will be saved to NFMs by smart FTLs. Hence, (1) needs to be updated as follows.

$$T_W = D \cdot T_{DE} \quad (2)$$

In our case, we also need to consider the data read time from the cache buffer. However, the cache buffer is usually implemented as DRAMs and their bandwidth is several orders of magnitude higher than that of NFMs. Furthermore, the slow NFM write operations hide most of the cache buffer read operations. For this reason, we omit the cache buffer read time in (1) for simplicity.

During the data transfer, the power consumption varies a lot as shown in Fig. 8. Fortunately, the power varies periodically, hence it is a good approximation to use the average value for a single period. P_{AVG}, average power of a multi-channel/multi-way SSD is given as follows.

$$P_{AVG} = I_{SC} \cdot V_{SC} + I_{DR} \cdot V_{DR} + N_c \cdot W \cdot (I_{NFM} \cdot V_{NFM}) \quad (3)$$

where I_{SC}, I_{DR}, and I_{NFM} are the currents of the SSD controller, DRAMs, and NFMs, respectively. Similarly, V_{SC}, V_{DR}, and V_{NFM} are their voltages, respectively. In the third term of (3), $N_c W$ represents the total number of NFMs.

From (1) and (3), the total energy consumed by SSD for data backup is as follows.

$$E_{TOTAL} = P_{AVG} \cdot T_W \quad (4)$$

The capacity of a backup power source should be larger than the energy computed by (4).

The remaining issue is to determine the peak capacity of the backup power source. In Fig. 8, the peak power consumption occurs when all the channels and ways are busy. To make all of them busy, the SSD controller and DRAMs should work at their maximum performance, too. For this reason, the peak power of multi-channel/multi-way SSD can be simply computed by replacing the average current components with maximum current components in (3).

4.5 Power Failure Notification and Data Backup

As discussed in Sects. 4.2 and 4.3, the power failure is notified to the SSD by PFDN. When this signal is activated, the SSD controller calls an interrupt service routine for data backup. This service routine is the same as the one which is called when the host machine is turned off through the normal power-off sequence. When the host machine is normally turned off, its operating system commands the device driver to send a shut-down command to the SSD. When the SSD receives this command, the SSD controller is interrupted for saving the updated data in volatile memories to NFMs. Even though the cause of the interrupt in our case is different from that of a normal power-off sequence, the service to be completed by the SSD controller is identical.

5. Experimental Results

5.1 Experimental Setting

To appreciate the efficiency of our method, we constructed two different experimental environments. First, we built a simulator for analyzing the internal behavior of the SSD when the power failure occurs. The architecture modeled in the simulator is 4-channel/2-way SSD. We implemented two PFDN boards to validate our method in a real notebook computer environment. One is super-capacitor based PFDN and the other is rechargeable battery based PFDN. We measured the stability of the system power supplied to the SSD and confirmed that the SSD safely backs up all the updated data residing in volatile memory.

5.2 Simulation-Based Analysis

In this analysis, we conducted a set of simulations for analyzing the impact of PFDN. The first simulation compares the performance of the write-back policy and the write-through policy. Without PFDN, it is not possible to use the write-back policy, since a sudden power-off will cause the loss of all updated data in volatile memories. Second, we measured the ratio of dirty pages to the size of the cache buffer when the write-back policy is employed. This simulation guides us to tune the capacity of the backup power source to back up the updated data in the volatile memories. As a result, we compared the battery capacity computed by (3) in Sect. 4.4 with the simulated power consumption when sudden power failure occurs. For this purpose, we prepared

Table 1 Traces used in simulation.

Trace	# of Request	Write/Read Length	Description
Distilled [13]	100,000	22.3 KB/ 39.6 KB	FAT32 / Various PC usage
Real Tr 1	46,942	22.3 KB/ 30.6 KB	NTFS / Windows 7 boot
Real Tr 2	37,731	52.1 KB/ 24.0 KB	NTFS / Small file copy
Real Tr 3	100,000	35.7 KB/ 50.5 KB	NTFS / General usage

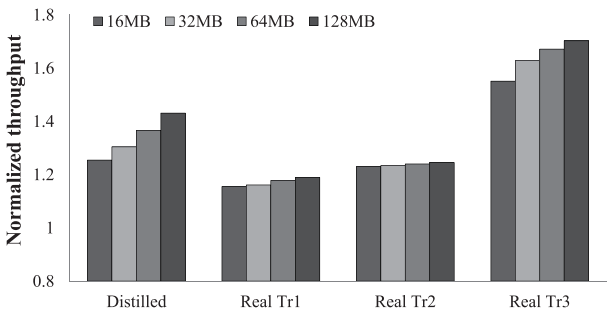


Fig. 9 The normalized throughput for write-back according to cache buffer size.

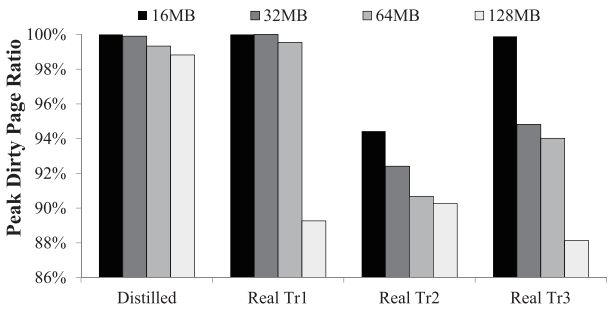


Fig. 10 The peak dirty page ratio.

a set of real traces as shown in Table 1.

Figure 9 shows the throughput of write-back policy normalized by that of write-through policy with respect to the cache buffer size, for the traces shown in Table 1. In this comparison, the write-back policy outperforms the write-through policy for all ranges of cache buffer size. Their performance gap becomes larger as the cache buffer size increases. For instance, the write-policy shows 1.7 times higher throughput over the write-through policy. This comparison clearly shows the necessity of PFDN from the performance perspective, since the write-back policy cannot be employed without PFDN for the sake of data reliability.

Next, we analyze the dirty page ratio over the cache buffer size to understand the portion of updated pages with respect to the cache buffer size. Depending on the access patterns, the dirty page ratio changes dynamically. For the backup power source, the worst case power failure occurs when the dirty page ratio is maximized. The backup power source capacity should be tuned for this scenario, otherwise, it will not provide sufficient power to the SSD if the power fails when the dirty page ratio peaks. Figure 10 compares

Table 2 Energy ratio for peak dirty pages.

Trace	E_{TOTAL} by (4)	E_{TOTAL} Simulation	Ratio between (4) and Simulation
Distilled	1.681 J	1.669 J	99.33%
Real Tr 1	1.681 J	1.673 J	99.55%
Real Tr 2	1.681 J	1.524 J	90.68%
Real Tr 3	1.681 J	1.580 J	94.02%

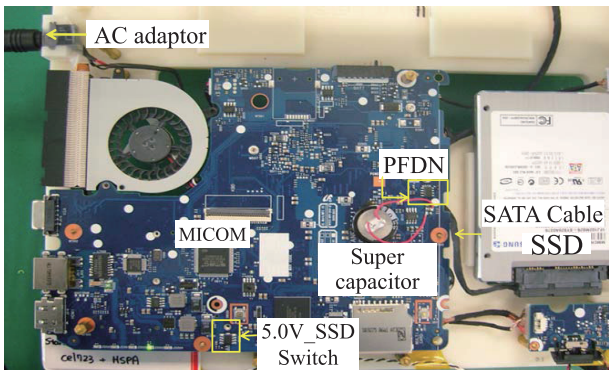


Fig. 11 Super-capacitor based PFDN plugged into a laptop computer equipped with an SSD.

the peak dirty page ratios for all traces, while changing the cache buffer size from 16 MB to 128 MB. For some traces, the peak dirty page ratio becomes 100%, especially when the cache buffer size is small. Even for the large cache buffer, the peak dirty page ratio is a little bit lower than 90%. Hence, it is not an overdesign to set D = 1.0 for determining the backup power source capacity.

Table 2 compares the backup power source capacity (E_{TOTAL}) computed by (4) and the simulated E_{TOTAL}. The statically determined E_{TOTAL} by (4) is not much different from the dynamically analyzed simulation result; hence, it is confirmed that (4) is efficient for determining the capacity of the backup power source.

To summarize, PFDN is an essential component for improving the performance of an SSD, since it allows us to employ the write-back policy for a cache buffer without sacrificing data reliability. Also, it is not an over-design to determine the backup power source capacity for the worst case scenario (whole cache buffer is filled with dirty pages), since the peak dirty page ratio in many real traces reaches 100%.

5.3 Board-Level Analysis

As mentioned in Sect. 5.1, we implemented both super-capacitor based PFDN and rechargeable battery based PFDN. Figure 11 shows the super-capacitor based PFDN integrated with a notebook computer which is equipped with an SSD. The rechargeable battery based PFDN can also be plugged into this system in the same manner.

In this experimental setting, we executed a program to write several hundreds of MB to the SSD to fill the SSD's internal cache buffer with dirty pages. As soon as the program was completed, we removed the power plug of the notebook

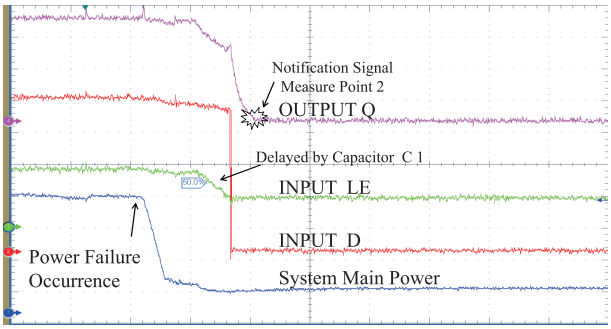
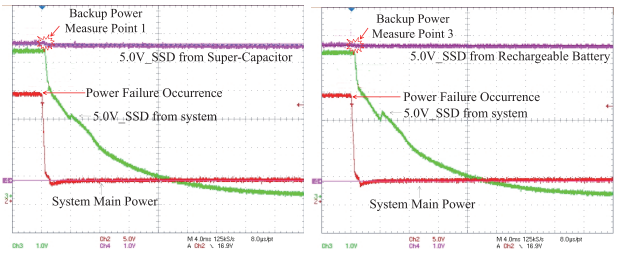


Fig. 12 Internal signals of super-capacitor based PFDN.



(a) (b)

Fig. 13 Measurement of PFDN on the note PC platform. (a) Super-capacitor based PFDN. (b) Super-capacitor based PFDN.

to validate the implemented super-capacitor based PFDN. Finally, we rebooted the notebook and read the SSD to confirm that all data were stored on the SSD.

We measured the variation in major signals of the super-capacitor based PFDN. It is shown in Fig. 12. Before the power failure, the system power is stably supplied to the SSD. After removing the power plug, the system power suddenly falls down and *INPUT_D* sharply goes to 'LOW'. Also, *INPUT_LE* goes to 'LOW' slightly after *INPUT_D*, hence the D-type latch successfully captures the change in *INPUT_D* and *OUTPUT_Q* becomes 'LOW'. The signal changes in Fig. 12 are exactly what we intended in Fig. 5. It takes about 1.5 ms from the power failure occurrence to the notification signal, which consumes about 2.52 mJ of energy. Considering Table 2, this delay consumes fairly negligible portion of the total backup power source.

At the same time, we measured the power supplied to the SSD, as shown in Fig. 13. As soon as the system power fails, the power supplied to the SSD from the system power decays continuously (noted as 5.0V_SSD from the system). However, the SSD can be stably operated, thanks to the super-capacitor, since it stably supplies the required power to the SSD. There are no power fluctuations in Backup Power Measure Point 1, which is the power node of the SSD in Fig. 5, which validates the stability of a super-capacitor based PFDN. The same experiment was performed for rechargeable battery based PFDN, and its power variation is shown in Fig. 13 (b). Figure 13 (b) shows results similar to Fig. 13 (a), meaning that the rechargeable battery based PFDN operated as we expected.

6. Conclusion

We propose two different power-failure protection schemes to increase the data reliability of SSDs. One is based on a super-capacitor, and the other is based on a rechargeable battery. We validated our method through a real board implementation as well as simulations. The simulations is effective for determination of backup power source capacity with over 90% of accuracy. On the other hand, in the real board implementations, both schemes successfully detect power failure and notify it to the SSD in negligible time of under 1.5 ms, without any power fluctuations. Finally, our method is a cost-effective solution, because the implementation of each scheme costs under one dollar.

Acknowledgments

This work was supported in part by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (No. 2011-0003559 and No. 2011-0027625) and by Samsung Electronics Company.

References

[1] G. Eskelsen, A. Marcus, and W.K. Ferree, The Digital Economy Fact Book, Tenth ed., The Progress & Freedom Foundation, Washington, D.C., 2009.
[2] J. Kim, J.M. Kim, S.H. Noh, S.L. Min, and Y. Cho, "A space-efficient flash translation layer for compact flash systems," IEEE Trans. Consum. Electron., vol.48, no.2, pp.366–375, May 2002.
[3] S.W. Lee, W.K. Choi, and D.J. Park, "FAST: An efficient flash translation layer for flash memory," Proc. 4th EUC Workshops, pp.879–887, Seoul, Korea, Aug. 2006.
[4] J.-U. Kang, H. Jo, J.-S. Kim, and J. Lee, "A superblock-based flash translation layer for NAND flash memory," Proc. 6th EMSOFT, pp.161–170, Seoul, Korea, Oct. 2006.
[5] H. Kim and S. Ahn, "BPLRU: A buffer management scheme for improving random writes in flash storage," Proc. 6th FAST, pp.239–252, San Jose, California, USA, Feb. 2008.
[6] S. Park, D. Jung, J. Kang, J. Kim, and J. Lee, "CFLRU: A replacement algorithm for flash memory," Proc. 7th CASES, pp.234–241, Seoul, Korea, Oct. 2006.
[7] H. Jo, J.-U. Kang, S.-Y. Park, J.-S. Kim, and J. Lee, "FAB: Flash-aware buffer management policy for portable media players," IEEE Trans. Consum. Electron., vol.52, no.2, pp.485–493, May 2006.
[8] S.-H. Park, S.-H. Ha, K. Bang, and E.-Y. Chung, "Design and analysis of flash translation layers for multi-channel NAND flash-based storage devices," IEEE Trans. Consum. Electron., vol.55, no.3, pp.1392–1400, Aug. 2009.
[9] P. Mars, "Supercapacitors for SSD backup power," Electronic Products, vol.51, no.10, pp.40–41, March 2009.
[10] K. Takeuchi, "Novel co-design of NAND flash memory and NAND flash controller circuits for sub-30 nm low-power high-speed solid-state drives (SSD)," IEEE J. Solid-State Circuits, vol.44, no.4, pp.1227–1234, April 2009.
[11] K. Takeuchi, Y. Kameda, S. Fujimura, H. Otake, K. Hosono, H. Shiga, Y. Watanabe, T. Futatsuyama, Y. Shindo, M. Kojima, M. Iwai, M. Shirakawa, M. Ichige, K. Hatakeyama, S. Tanaka, T. Kamei, J.Y. Fu, A. Cernea, Y. Li, M. Higashitani, G. Hemink, S. Sato, K. Oowada, S.-C. Lee, N. Hayashida, J. Wan, J. Lutze, S. Tsao, M. Mofidi, K. Sakurai, N. Tokiwa, H. Waki, Y. Nozawa, K. Kanazawa,

and S. Ohshima, "A 56-nm CMOS 99-mm^2 8-Gb multi-level NAND flash memory with 10-MB/s program throughput," IEEE J. Solid-State Circuits, vol.42, no.1, pp.219–232, Jan. 2007.
[12] S.K. Won, S.-H. Ha, and E.Y. Chung, "Fast performance analysis of NAND flash-based storage device," Electron. Lett., vol.45, no.24, pp.1219–1221, Nov. 2009.
[13] L.-P. Chang and T.-W. Kuo, "An adaptive striping architecture for flash memory storage systems of embedded systems," Proc. 8th RTAS, pp.187–196, San Jose, California, USA, Sept. 2002.

Kwanhu Bang received the B.S. degrees in computer science and in electronic engineering and the M.S. degree in electrical and electronic engineering from Yonsei University, Seoul, Korea, in 2006 and 2008, respectively. He is currently a Ph.D. candidate in the School of Electrical and Electronic Engineering at Yonsei University. His research interests include flash memory applications, system-level low-power design and bio-computation.

Kyung-Il Im received the B.S. degree in electrical and electronic engineering from Inha University in Korea, in 2001, and M.S. degree in electrical and electronic engineering from Yonsei University, Seoul, Korea, in 2011. He is currently an engineer with Computer Systems Division, Samsung Electronics in Suwon-City, Gyeonggi-Do, Korea. His research interests include System on Chip, NAND flash based mass storage architecture and system architecture.

Dong-gun Kim received the B.S. and M.S. degree in electrical and electronic engineering from Yonsei University, Seoul, Korea, in 2010 and 2012, respectively. He is currently an engineer with Advanced Design System Architecture Team, SK Hynix Semiconductor Inc. in Icheon-si, Gyeonggi-Do, Korea. His research interests include System on Chip, NAND flash based mass storage architecture and system architecture.

Sang-Hoon Park received the B.S. degree in electrical and electronic engineering from Yonsei University in Seoul, Korea, in 2009. He is currently a Ph.D. candidate in Yonsei University. His research interests include System on Chip, NAND flash based mass storage architecture and system architecture.

Eui-Young Chung received the B.S. and M.S. degrees in electronics and computer engineering from Korea University, Seoul, Korea, in 1988 and 1990, respectively, and the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, in 2002. From 1990 to 2005, he was a Principal Engineer with SoC R&D Center, Samsung Electronics, Yongin, Korea. He is currently an Professor with the School of Electrical and Electronic Engineering, Yonsei University, Seoul, Korea. His research interests include system architecture, bio-computing, and VLSI design, including all aspects of computer-aided design with the special emphasis on low power applications and flash memory applications.