Using Dynamic Voltage Scaling for Energy-Efficient Flash-based Storage Devices

Sungjin Lee and Jihong Kim
School of Computer Science and Engineering, Seoul National University, Korea
{*chamdoo, jihong*}*@davinci.snu.ac.kr*

ABSTRACT

NAND flash memory is commonly known as a power-efficient storage medium. Because of the increasing complexity of flash-based storage devices, however, it is more difficult to achieve good power-efficiency without considering an energy-efficient storage device design. In this paper, we investigate the potential benefit of dynamic voltage/frequency scaling (DVFS) on the energy-efficiency of flash-based storage devices. We first develop a performance/power model for a flash device by using an FPGA-based flash device platform. We then propose a simple DVFS heuristic algorithm that exploits workload fluctuations of a flash device to achieve a significant reduction in energy consumption without performance degradation. Experimental results show that a flash device with DVFS can reduce energy consumption by up to 20%-30%.

1. INTRODUCTION

Unlike hard disk drives, flash-based devices do not require any mechanical parts, thus providing higher power-efficiency and better performance than traditional storage devices. Unfortunately, as the density of flash devices increases, the performance and reliability of flash memory cells are significantly deteriorated. Furthermore, the increasing popularity of the multi-level cell (MLC) technology, which is inferior in terms of performance and reliability, also makes these problems more critical. To cope with such difficulties, specialized hardware and software supports are considered essential in designing flash devices.

However, the increasing complexity of hardware and software components causes flash-based storage devices to consume more energy than ever before, lowering the overall energy-efficiency. For instance, Seo et al. [1] claimed that some flash-based solid-state disks (SSDs) dissipate more energy than conventional hard disk drives, especially when they stay in an idle state or perform random writes. A similar observation [2] shows that the energy saving from using SSDs is not significant; some SSDs even exhibit poorer energy-efficiency than hard disk drives. These observations strongly motivate the need for power-efficient designs for flash-based storage devices.

In this paper, we investigate the effectiveness of dynamic voltage/frequency scaling (DVFS) in building a low-power flash device. We first build a performance/power model for flash devices using FPGA-based flash device designs. By using this model, we develop a flash device emulator that supports the DVFS capability. Finally, we propose a heuristic DVFS algorithm and then evaluate its effectiveness in terms of performance and energy. Our experimental results show that the energy consumption of a flash device can be reduced by up to 20%-30% with the proposed DVFS algorithm.

This paper is organized as follows. Section 2 briefly reviews low-power studies for flash devices. In Section 3, we explain the overall architecture of our flash storage platform.

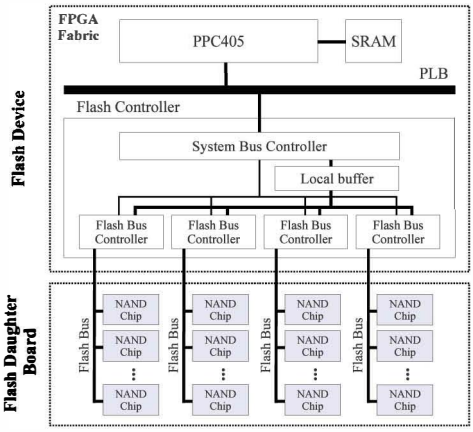

Figure 1: Overall architecture of our flash platform

In Section 4, we analyze the performance/power characteristics of a flash device at varying supply voltage levels. Section 5 describes the proposed DVFS algorithm in detail. The experimental results are given in Section 6. Section 7 concludes with a summary and proposals for future work.

2. RELATED WORK

There have been not many research studies relevant to power consumption of a flash device. Seo et al. [1] analyzed the power characteristics of flash-based SSDs under a variety of I/O access patterns, but proposed no power optimization techniques. Park et al. [3] presented a simple power model for flash-based SSDs using real power measurements, and then evaluated the effectiveness of several DPM policies redesigned for SSDs. Our approach is quite different from these works in that we consider a performance/power trade-off on a flash device with dynamic voltage/frequency scaling.

There are also approaches [4, 5] exploiting a NOR/NAND flash part that supports several voltage levels. They enable a flash chip to operate at a low-voltage level depending on the workload and achieve effective power reduction while providing reasonable performance. The major drawback of these approaches, however, is that they cannot be used with recent flash parts because most flash chips usually support a single voltage level. On the other hand, our work takes into account the design of a low-power flash storage from system-level perspectives and is thus easily applicable to various flash devices including mobile embedded systems and flash-based SSDs.

3. A TARGET FLASH PLATFORM

Figure 1 shows the overall architecture of flash-based storage devices. Our target flash device is BlueSSD [6], an open flash platform which is based on the Zarkov-I SSD prototype

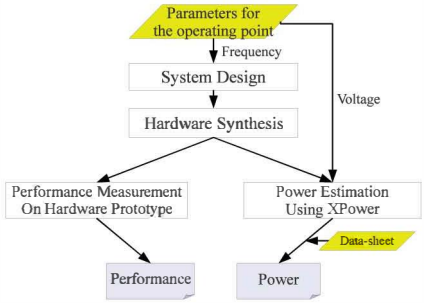

Figure 2: Steps of performance/power estimation

developed at UCSD [7]. BlueSSD is completely implemented on an FPGA fabric, thus allowing us to easily reconfigure its hardware components. The software framework of BlueSSD also provides a set of APIs which make it easy to assess the performance of the hardware system.

There are three major modules in the BlueSSD hardware: a microprocessor, a flash daughter board, and a flash controller. The microprocessor is responsible for executing flash software, such as a flash file system and a flash translation layer (FTL), and issues I/O commands like a page read/write and a block erasure to the flash controller. The microprocessor we have used is IBM's PowerPC405 (PPC405), a 400MHz hard IP processor provided by Xilinx FPGA platforms. The flash daughter board is an array of NAND flash chips on a custom PCB board. A flash chip has a number of control lines (e.g., ALE, CLE, and CE) and an 8-bit flash bus. The daughter board is designed to support four pairs of control lines and four flash buses so that four flash chips can be accessed in parallel. The flash controller allows us to transfer data from or to the flash daughter board and is implemented as a peripheral for IBM's Processor Local Bus (PLB). The PLB provides 32-bit address and 64-bit data widths, and is set by default to be working at 100MHz. Our flash controller is also composed of two sub modules. The first is the system bus controller which accepts I/O commands from the processor and then performs data transfers from SRAM to the local buffer. It has been designed to work at the same frequency with the PLB. Second, the flash bus controller is in charge of transferring data between the local buffer and the flash daughter board through flash buses. It also works in sync with the frequency of a flash bus, which is generally operated at the maximum of 33 MHz. To exploit the parallelism by multiple flash buses, there are four bus controllers in our implementation.

4. PERFORMANCE AND POWER MODEL FOR A FLASH DEVICE WITH DVFS

4.1 Performance/Power Estimation Methodology

The primary goal of our work is to understand potential benefits of adopting DVFS techniques for a flash-based device. To accomplish this goal, the performance/power characteristics should be accurately estimated according to the operating point which represents a supply voltage and a working frequency of each component.

Figure 2 summarizes the overall steps of the performance and power estimation. We first make a flash device design that operates at the frequency of the given operating point and synthesize it to produce a hardware prototype. This is possible because our FPGA platform provides a clock manager which allows designers to determine the frequency of system components at the design time. In order to know the performance at the given operating point, we then measure the time taken to perform I/O operations (e.g., page reads, page writes, and block erasures) on the hardware prototype. Unfortunately, our FPGA platform does not provide multiple voltage levels, and it is thus impossible to measure the power dissipation for the operating points on a real hardware. Instead, the power consumption of the design is obtained through a commercial power analysis tool called XPower [8]. XPower can estimate the power consumption of several hardware components such as clock trees, logics, signals, hard IPs, and I/Os by utilizing device knowledge and design data. In order to include the power consumed by flash chips, we have used the values specified on the datasheet of a flash part.

	Frequency (MHz)				Voltage (V)	
	PPC405	PLB/Sys. Bus Ctrl.	Flash Bus Ctrl.	Relative Freq. (Φ_i)	V_{logic}	V_{io}
OP_1	100	100	33	1.00	1.2	3.3
OP_2	66	66	22	0.66	1.1	3.3
OP_3	50	50	17	0.50	1.0	3.3
OP_4	33	33	11	0.33	1.0	3.3

Table 1: Parameter settings for the operating points

Table 1 shows four operating points we are considering in the paper. The parameters used here are borrowed from PPC405-LP [9], which is based on the PPC405 core and is modified to support dynamic voltage and frequency scaling. V_{logic} is the voltage applied to all of the logic components including PPC405, and V_{io} is the voltage for I/O pins of flash chips. V_{io} should be kept at 3.3V [10]. The difference between the minimum and maximum voltages of V_{logic} is set to 0.2V to reflect a narrow voltage scaling range of a modern microprocessor. There are three clock domains in the current design: PPC405, a PLB/system bus controller, and a flash bus controller. PPC405 and PLB are located on different clock domains, but they are set to operate at the same clock for simplicity. The frequency of the system bus controller is the same as that of PLB because it is designed to be synchronized with PLB. The flash bus controller is set to slow down in proportion to other components. The relative frequency (Φ_i) represents the ratio of the frequency of OP_i to the highest frequency of OP_1.

4.2 Performance/Power Model

Table 2(a) shows the elapsed time taken to perform page read, page write, and block erase operations depending on the operating points. The time is categorized into two types: the time taken by the flash device and the time spent by the NAND flash chip. The former indicates the time during which the flash device handles I/O requests so that data is read from or written to the NAND flash chip. The latter is the time during which the flash chip performs its internal op-

	Flash Device				NAND Chip
	OP_1	OP_2	OP_3	OP_4	
Read (page)	79	132	161	250	40
Write (page)	84	138	165	246	220
Erase (block)	8	10	11	14	2025

(a) Performance measurement results (μs)

		Flash Device				NAND Chip
		OP_1	OP_2	OP_3	OP_4	
Active	Logic/Sig./Clk.	1566	854	535	354	40
	IO	302	199	151	100	40
	Quiescent	50	50	50	50	40
Idle		224	153	113	95	3

(b) Power estimation results (mW)

	Flash Device				NAND Chip
	OP_1	OP_2	OP_3	OP_4	
Read (page)	151	145	118	125	1.6
Write (page)	157	152	121	123	8.8
Erase (block)	12	9	7	5	81

(c) Energy consumption of each I/O operation (μJ)

Table 2: Performance/power estimation results

erations such as transferring data between the on-chip data register and memory cells as well as erasing a flash block. As shown in Table 2(a), there is no change in the time taken for these internal operations because they are not influenced by the frequency scaling on the flash device. Therefore, the time taken by the NAND chip is constant regardless of operating points. On the other hand, the performance of the flash device is noticeably reduced as the frequency decreases. Based on this observation, the page write time $T_w(OP_i)$ at the operating point OP_i can be expressed as follows:

$$T_w(OP_i) = T_w^{dev}(OP_i) + T_w^{chip}$$
$$\text{where } T_w^{dev}(OP_i) = T_w^{dev}(OP_1) \cdot (1/\Phi_i),$$

where $T_w^{dev}(OP_1)$ and $T_w^{dev}(OP_i)$ are the write times of the flash device at the operating points OP_1 and OP_i, respectively. T_w^{chip} is the time taken by the flash chip when writing a page. For example, $T_w^{dev}(OP_1)$ and T_w^{chip} are 84μs and 220μsec in Table 2(a), respectively. Note that the page read time and the block erasure time can be defined in a similar manner.

Table 2(b) shows the power consumption according to the type of power-dissipation sources when the activity rate is set to 0.75. As expected, the power consumption of the logic/signal/clock is proportional to $V^2 * f$. However, the I/O power is determined by the working frequency f because the applied voltage V_{io} is set to the fixed 3.3V. The quiescent is the power consumed with no signals switching [8]. In this work, the power dissipation $P_{rw}^{dev}(OP_i)$ observed when the flash device performs read and write operations is defined as the sum of the power consumption of all the components operated at the operating point OP_i. For example , $P_{rw}^{dev}(OP_1)$ is 1918 mW in Table 2(b). However, the I/O power is excluded for an erase operation because there is no data transfer through I/O pins during block erasure operations. The power consumption P^{chip} of the flash chip is from Samsung's K9F8G08UXM data-sheet. The proportion of the flash chip among total power consumption is trivial. The power dissipation which occurs when the device is idle is less significant than the active state, but there is a meaningful power difference between operating points. Finally, the energy consumption $E_w(OP_i)$ for a write operation at the operating point OP_i can be expressed as follows:

$$E_w(OP_i) = T_w^{dev}(OP_i) \cdot P_{rw}^{dev}(OP_i) + T_w^{chip} \cdot P^{chip}.$$

The energy consumed by page read and block erasure operations can be defined in the same way.

Table 2 displays the energy consumption of each I/O operation based on Table 2(a) and (b). This clearly shows that, in the best case, about 22% less energy is required to perform the same operation. However, the energy saving for an erase operation is trivial because the majority of the time is spent by the flash chip. In particular, the energy consumption of OP_4 is slightly higher than that of OP_3. This is mainly due to the increased proportion of the quiescent power.

Before finishing this section, we need to mention the performance penalty caused by changing the operating point from one to another. Our flash controller is built as a peripheral for PLB. According to [9], we can safely assume that the delay for voltage/frequency changes will be limited to the range of several ten microseconds (e.g., 13μs-95μs). In our work, this latency is set to 100μs for the worst case.

5. A HEURISTIC DVFS POLICY

To investigate potential benefits of DVFS on a flash device, we have developed a practical DVFS heuristic. Similar to typical DVFS algorithms, our heuristic is devised to achieve energy savings with little performance overhead by

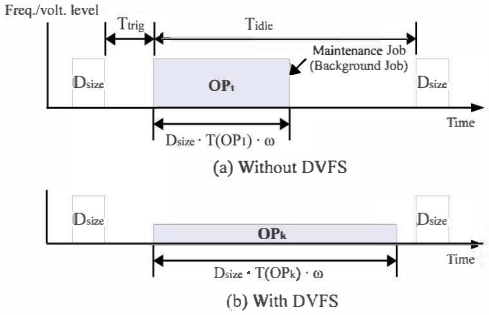

Figure 3: Overall approach of the DVFS algorithm

lowering the voltage level depending on the workload. Especially, we have focused on minimizing the energy consumed by extra I/Os involved in managing flash devices. Due to the physical limitations of NAND flash such as erase-before-write and limited erasure cycles, a flash-based device typically requires some maintenance jobs like garbage collection and wear-leveling which move a large amount of data from its original location to a new one. These operations incur many extra requests to the device, along with user I/O requests. For instance, Hu et al. [11] claimed that the ratio of extra I/Os over ordinary I/Os could be as large as 4.0 in the worst case. Note that this ratio is also called the write amplification factor (WAF). As a result, the energy dissipated by these extra I/Os could account for a significant proportion of total energy consumption, so it is important to deal with them effectively for an energy-efficient storage device.

In general, these extra I/Os are scheduled and performed in the background, so as to prevent user I/O requests from being undesirably delayed. Our goal is to adjust the voltage level (or the operating point) as low as possible for energy savings, but keep it high enough so that all the maintenance jobs scheduled can be processed within a given idle interval. Figure 3(a) explains a basic concept of the proposed DVFS algorithm. D_{size} is the amount of data read from or written to the flash and its size is the same as the block size of a file system. T_{idle} is the length of idle times until the following request is issued. If the value, ω, of WAF is given, the amount of data that should be moved during T_{idle} can be expressed as $D_{size} \cdot \omega$. Further suppose that $T_{mig}(OP_1)$ is the time taken to move data of D_{size} at the highest speed. In this case, $D_{size} \cdot T_{mig}(OP_1) \cdot \omega$ is the time required for the maintenance jobs. If there is a set of operating points $\mathbb{OP} = \{OP_1, ..., OP_n\}$, our objective is to choose the operating point OP_k with the lowest voltage/frequency which satisfies the condition $\{D_{size} \cdot T_{mig}(OP_k) \cdot \omega\} \leq T_{idle}$ as shown in Figure 3(b).

To realize this strategy on a real system, the proposed DVFS policy is implemented as two tasks: a foreground task and a background task. The foreground task is responsible for handling I/O requests from users and applications. In order to provide the highest performance for end-users it is always running at the highest operating point OP_1. If the idle time is detected and there are maintenance jobs, the background task processes them during the idle time. Unlike the foreground task, the background task has several operating points. As mentioned before, the foreground task selects the appropriate point depending on the idle time T_{idle} and the amount of data $D_{size} \cdot \omega$ to be moved. T_{idle} is obtained as a weighted average of the inter-arrival time between two successive I/O requests. Similarly, the value of ω is estimated using the average amount of extra I/O traffics which are gathered every 30 seconds. Since the value of D_{size} is fixed at a constant value, $D_{size} \cdot \omega$ can easily be calculated. The transition between the operating points, however, incurs the undesirable latency. To avoid this seri-

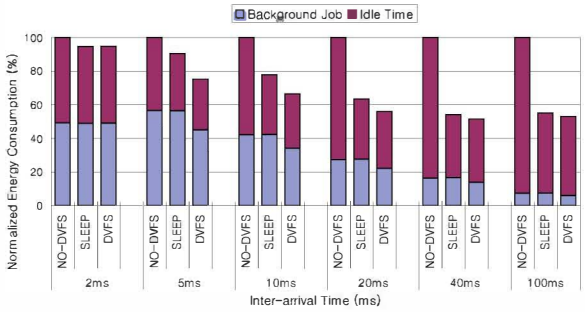

Figure 4: Normalized energy consumption

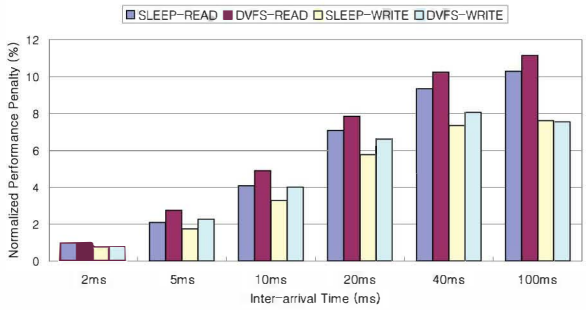

Figure 5: Normalized performance penalty

ous performance penalty, the background task needs to be activated when the idle time is larger than a certain threshold T_{trig}. If there are no maintenance jobs available for the background task, it is preempted by the idle task or other system tasks, thereby changing the state of the device to the lowest operating point OP_4 to minimize the energy dissipation in an idle state.

6. EXPERIMENTAL RESULTS

To evaluate the performance of the proposed DVFS algorithm, we have developed a flash device emulator which is designed to simulate the performance/power model mentioned in Section 4. This emulator was built as a block device driver for Linux kernel, so we can implement various DVFS algorithms on system software and assess their effects on performance and power. In our work, the DVFS algorithm was implemented on a flash file system, but it can be easily realized on other flash management software like FTL. The flash memory model used in our evaluation was based on Micron's NAND flash memory [10], whose capacity is 1GB and is composed of 64 2KB pages. Our emulator was organized to have four flash parts with four flash buses.

Regarding the parameters for the DVFS policy, D_{size} was set to 8KB and T_{trig} was 5ms. As for the evaluation, we modified the file system so that WAF is to be 1.0. The foreground task was set to use OP_1 as shown in Table 1. The background task supported three operating points OP_1, OP_2, and OP_3. To understand the effectiveness of the proposed DVFS policy across a spectrum of various conditions, we evaluated it under a Pareto distribution whose inter-arrival time is varied from 2ms to 100ms. There are three kinds of configurations: NO-DVFS, SLEEP, and DVFS. NO-DVFS uses no power optimization technique. SLEEP puts the device into OP_4 when the idle interval is larger than T_{trig}. This does not utilize the ability of processing I/O operations at the low-power state. DVFS refers to our DVFS algorithm.

Figure 4 shows the energy consumption when performing background (maintenance) jobs and when the device stays in the idle state. This result is normalized to NO-DVFS. When the inter-arrival time is too short, SLEEP and DVFS cannot achieve significant energy savings because there is little idle time whose length is long enough so that the device enters the low-power state. As the inter-arrival time increases, SLEEP reduces the energy wasted during the idle time and the benefit increases as the length of the idle period gets longer. DVFS reduces the energy dissipated by the background job as well as the energy used during the idle time. Especially, even when the inter-arrival time is relatively short, it reduces 10%-20% more energy than SLEEP by performing the background jobs in the low-power state. Figure 5 shows the normalized performance overhead caused by SLEEP and DVFS over NO-DVFS. As expected, as the length of inter-arrival time becomes longer, the performance penalty noticeably increases. Naturally, this is mainly because the device goes into the low-power state more frequently. However, this performance penalty is gradually reduced as the idle time is greater than 40ms.

7. CONCLUSION

In this paper, we have evaluated the effectiveness of dynamic voltage and frequency scaling (DVFS) in a flash-based storage device. For this purpose, we have made the performance/power model from an FPGA-based flash device. We also proposed a simple DVFS algorithm that exploits a flash device with DVFS. Our evaluation results show that DVFS is an effective approach in reducing the energy consumption of a flash-based device, even when the length of the idle time is relatively short. As future works, we plan to devise an accurate performance/power model for a flash-based device. We will also develop a more sophisticated DVFS algorithm which can accomplish more energy savings with little performance penalty.

8. ACKNOWLEDGMENTS

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (No. 20100018873, No. R33-2010-10095). The ICT at Seoul National University and IDEC provided research facilities for this study.

9. REFERENCES

[1] E. Seo et al., "Empirical Analysis on Energy Efficiency of Flash-based SSDs," in *Proceedings of the International Workshop on Power Aware Computing and Systems*, 2008.

[2] The SSD Power Consumption Hoax, `http://www.tomshardware.com/reviews/ssd-hdd-battery,1955.html`, 2008.

[3] J. Park et al., "Power Modeling of Solid State Disk for Dynamic Power Management Policy Design in Embedded Systems," in *Proceedings of the International Workshop on Software Technologies for Embedded and Ubiquitous Systems*, pp. 24-35, 2009.

[4] L. Chang et al., "A Dynamic-Voltage-Adjustment Mechanism in Reducing the Power Consumption of Flash Memory for Portable Devices," in *Proceedings of the Internation Conference on Consumer Electronics*, pp. 218-219, 2001.

[5] Y. Du et al., "Dynamic Voltage Scaling of Flash Memory Storage Systems for Low-Power Real-Time Embedded Systems," in *Proceedings of the International Conference on Embedded Software and Systems*, pp. 152-157, 2005.

[6] S. Lee et al., "BlueSSD: An Open Platform for Cross-layer Experiments for NAND Flash-based SSDs," in *Proceedings of the International Workshop on Architectural Research Prototyping*, 2010.

[7] A. M. Caulfield et al., "Gordon: Using Flash Memory to Build Fast, Power-Efficient Clusters for Data-Intensive Applications," in *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 217-228, 2009.

[8] Xilinx Corp., "XPower Estimator User Guide," `http://www.xilinx.com/support/documentation/user_guides/ug440.pdf`.

[9] B. Brock et al., "Dynamic Power Management for Embedded Systems," in *Proceedings of the International SoC Conference*, pp. 416-419, 2003.

[10] Micron Corp., "2,4,8Gb: x8/x16 Multiplexed NAND Flash Memory Features," 2004.

[11] X. Hu et al., "Write Amplification Analysis in Flash-based Solid State Drives," in *Proceedings of the Israeli Experimental Systems Conference*, 2009.