Understanding the Impact of Power Loss on Flash Memory

Hung-Wei Tseng Laura M. Grupp Steven Swanson
The Department of Computer Science and Engineering
University of California, San Diego
{h1tseng,lgrupp,swanson}@cs.ucsd.edu

Abstract

Flash memory is quickly becoming a common component in computer systems ranging from music players to mission-critical server systems. As flash plays a more important role, data integrity in flash memories becomes a critical question. This paper examines one aspect of that data integrity by measuring the types of errors that occur when power fails during a flash memory operation. Our findings demonstrate that power failure can lead to several non-intuitive behaviors. We find that increasing the time before power failure does not always reduce error rates and that a power failure during a program operation can corrupt data that a previous, successful program operation wrote to the device. Our data also show that interrupted program operations leave data more susceptible to read disturb and increase the probability that the programmed data will decay over time. Finally, we show that incomplete erase operations make future program operations to the same block unreliable.

Categories and Subject Descriptors

B.3.4 [**Memory Structures**]: Reliability, Testing, and Fault-Tolerance

General Terms

Experimentation,Measurement,Performance,Reliability

Keywords

flash memory, power failure, power loss

1. INTRODUCTION

As flash-based solid-state drives (SSDs) make inroads into computer systems ranging from data centers to sensor networks, the integrity of flash memory as a storage technology becomes increasingly important. A key component of that integrity is what happens to the data on an SSD when power failure occurs unexpectedly.

Power loss in flash is potentially much more dangerous than it is for conventional hard drives. If power fails during a write to a hard drive, the data being written may be irretrievable, but the other data on the disk remains intact. However, SSDs use complex flash translation layers (FTLs) to manage the mapping between logical block addresses and physical flash memory locations. FTLs must store metadata about this mapping in the flash memory itself, and, as a result, corruption of the storage array can potentially render the entire drive inoperable: Not only will the in-progress write not succeed, but all the data on the drive may become inaccessible.

To ensure reliability, system designers must engineer the SSDs to withstand power failures and the resulting data corruption. To do this, they must understand what kinds of corruption power failure can cause.

This paper characterizes the effect of power failure on flash memory devices. We designed a testing platform to repeatedly cut power to a raw flash device during program and erase operations. Our data show that flash memory's behavior under power failure is surprising in several ways. First, operations that come closer to completion do not necessarily exhibit fewer bit errors. Second, power failure not only results in failure of the operation in progress, it can also corrupt data *already* present in the flash device. Third, power failure can negatively impact the integrity of future data written to the device. Our results point out potential pitfalls in designing flash file systems and the importance of understanding failure modes in design embedded storage systems.

The rest of this paper is organized as follows: Section 2 describes the aspects of flash memory pertinent to this study. Section 3 describes our experimental platform and methodology for characterizing flash memory's behavior during power failure. Section 4 presents our results and describes the sources of data corruption due to power failure. Section 5 provides a summary of related work to put this project in context, and Section 6 concludes the paper.

2. FLASH MEMORY

Flash memory has several unique characteristics that make power failure particularly dangerous. The long latency of program and erase operations present large "window of vulnerability" and the complex programming algorithms and data encoding schemes they employ can lead to non-intuitive failure modes. This section presents a summary of flash's characteristics that are most pertinent to this work.

Flash memory stores data by trapping electrons using a floating gate transistor. The electrons affect the transistor's threshold voltage, and the chip measures this change to read data from the cell. The flash chip organizes cells into pages (between 2 and 8 KB) and pages into blocks (between 32 and 256 pages). Erasing a block sets all the bits to '1'. Finer grain erasure is not possible. Programs operate on pages and convert 1s to 0s. To hide the difference in granularity between programs and erases and increase reliability, SSDs use complex flash translation layers (FTLs) to perform out-of-place update and remapping operations. To support these functions, FTLs store metadata in the flash storage array along with the user data.

Program and erase operations are iterative. Program operations selectively inject electrons into floating gates to change the threshold voltage and then perform a read-verify operation to check if the

DAC 2011, June 5-10, 2011, San Diego, California, USA.

Voltage Levels	Logic Bits			
	Gray coding		2's complement coding	
	1st page bit	2nd page bit	1st page bit	2nd page bit
Lowest	1	1	1	1
	1	0	1	0
	0	0	0	1
Highest	0	1	0	0

Table 1: The mapping of voltage level and logic bits in 2-bit MLC chips using Gray coding and 2's complement coding.

cells have reached the desired threshold voltage. If any of the cells in a page has not reached the target threshold voltage, the chip will repeat the program and read-verify process [15, 6]. For erase operations, the chip removes the electrons from cells within the block. The chip will continue to remove electrons until the voltages of cells reach the erased state.

There are two types of flash cells: single-level cell (SLC) and multi-level cell (MLC). SLC devices store one bit per cell, while MLC devices store two or more. SLC chips provides better and more consistent performance than MLC chips. According to empirical measurements in [3] it takes an SLC chip 20 μs to perform a read operation, 200 μs to perform a program operation, and 400 μs - 2 ms to perform an erase operation.

MLC chips achieve higher densities by using 2^n threshold voltage levels to represent n bits. MLC chips need 300 μs - 2 ms to perform a program operation, and 2 ms – 4 ms to perform an erase operation. In this paper, we focus on 2-bit MLC cells, since they are most prevalent in current systems.

For 2-bit MLC chips, cells store data of two different pages. Manufactures require that pages within a block be programmed in order, so to differentiate between the two pages in a cell, we refer to them as "first page" and "second page." Programming a second page is consistently slower than programming a first page, since programming the second page requires a more complex programming algorithm. Table 1 shows the mappings between threshold voltages and logic bits of a 2-bit MLC cell using gray coding and 2's complement coding.

3. METHODOLOGY

To study the effect of power failure on flash memory, we built a test platform that allows us to issue command to raw flash chips and to cut off the power supply to the chip at precise moments. This section describes our test bed, testing methodology, and the flash chips we used in this study.

3.1 Experimental hardware

For this work, we built a test platform that consists of three components: the Xilinx XUP board, a custom flash testing board, and the power control circuit.

The FPGA on the Xilinx XUP board contains a PowerPC 405 core running Linux. A custom flash controller on the FPGA provides us direct access to the flash device via the flash testing board. The FPGA also controls the power to the flash chips by means of a pair of high-speed power transistors. Measurements with an oscilloscope show that the system can switch the chip's power supply to 0 V within 3.7 μs.

3.2 Test procedure

To test the impact of power failure during program and erase operations, we cut power to flash at different points during the operation. We define the *power cut off interval* as the time between issuing the command to the flash chip and when we trigger the power cut off circuit. We start the cut off interval after sending the last byte of the command to the flash chip. High-resolution measurements of the chips' power consumption show that the chip starts executing the command with a few microseconds.

For program tests, we use cut off intervals varying from 0.4 μs to 2.4 ms at increments of 0.4 μs. For erase, we use power cut off intervals varying from 2 μs to 4.8 ms at increments of 2 μs.

Abbrev.	Manufacturer	Cell Type	Cap. (GBit)	Tech. Node (nm)	Page Size (B)	Pgs/ Blk
A-SLC2	A	SLC	2		2048	64
A-SLC4	A	SLC	4		2048	64
A-SLC8	A	SLC	8	60	2048	64
B-SLC2	B	SLC	2	50	2048	64
B-SLC4	B	SLC	4	72	2048	64
E-SLC8	E	SLC	8		2048	64
A-MLC16	A	MLC	16		4096	128
B-MLC32-2	B	MLC	32	34	4096	256
D-MLC32	D	MLC	32		4096	128
E-MLC8	E	MLC	8		4096	128
F-MLC16	F	MLC	16	41	4096	128

Table 2: Parameters for the 11 flash devices we studied in this work

3.3 Flash devices

The behavior of flash memory chips from different manufacturers varies because of architectural differences within the devices and because of differences in manufacturing technologies. To understand the variation in power failure performance, we selected 11 chips that cover a variety of technologies and capacities.

Table 2 lists the flash memory chips that we studied in this work. They come from five different vendors. Their capacities range from 2 GBits to 32 GBits and their feature sizes range from 72 nm to 34 nm. Values that are not publicly available from the manufacturer are from [3].

4. EXPERIMENTAL RESULTS

We found unexpected behavior for both program and erase operations in the presence of power failure. For both program and erase, the variation in bit error rate as the power cut off interval changes is non-monotonic, and our measurements show that power loss can lead to both immediate and long-term data integrity issues. We describe the results for each operation in turn.

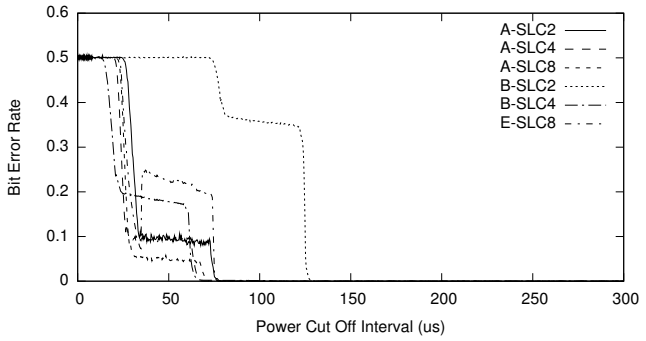
4.1 Program and power failure

To understand the impact of power failure during programming, we begin by programming random data and cutting off power at different intervals. Then, we measure the resulting bit error rate. Figure 1 contains the results for SLC chips (a) and MLC chips (b).

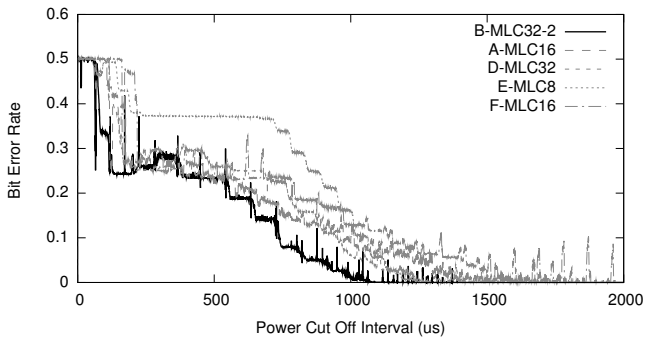
Intuitively, the more time we give the flash chip to program a page before power failure, the fewer errors there should be. However, the graphs show that the bit error rate does not decrease monotonically. Instead, the bit error rate for each chip has multiple plateaus – where the error rate remains constant, and spikes – where the bit error rate increases briefly.

For example, the error rate for E-SLC8 jumps dramatically at 30 μs, drops slowly until 75 μs when it plummets to nearly zero. The other SLC chips exhibit much more predictable behavior.

MLC behavior is much more complex. For example, B-MLC32-2's error rate remains constant at 50% until 100 μs and then drops sharply to 25% by 110 μs, where it remains until 200 μs. The error rate starts increasing at 200 μs and reaches 29% at 290 μs.



(a)



(b)

Figure 1: The bit error rate of program operations with different power cut off intervals for (a) SLC chips and (b) MLC chips

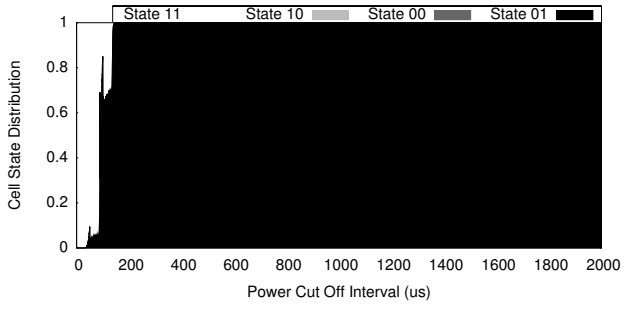
The error rate decreases again after 360 μs and then stays at 25% until 500 μs. After 500 μs, the error rate decreases as steps until it reaches 0 at 1400 μs. The chip also shows numerous spikes in error rate, for example, at 540 μs.

These results are unexpected because programming flash chips should only be able to move bits from 1 to 0, yet the non-monotone error rates suggest that program operations are moving cells in both directions at different times. Below, we investigate this behavior in finer detail.

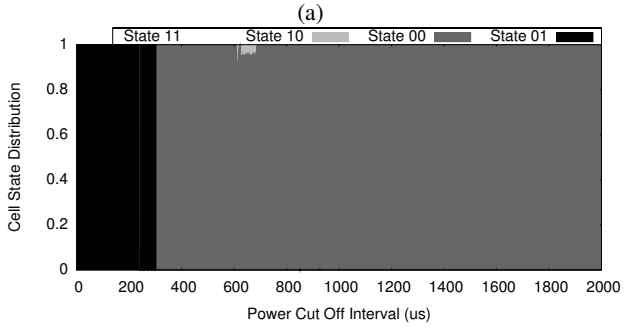
4.1.1 Per-page MLC error rates

To understand the cause of MLC chips' non-monotonic error rates, we examine the behavior of pairs of first and second pages in more detail. A pair of corresponding bits from the two pages can be in four states: 01 (i.e., the first page bit is '0' and the second page bit is '1'), 00, 11, and 10. We used program operations to move the pair between those states and interrupted the power to see which intermediate states the cell passes through. We consider four transitions: (1) 11→01: we program the first page bit to 0 from the erased state. (2) 01→00: we program a 0 to the second page bit after programming a 0 to the first page bit. (3) 01→01: we program a 1 to the second page bit after programming a 0 to the first page bit. (intuitively, this should cause no change) (4) 11→10: we program a 0 to the second page bit from the erased state. Other transitions are not possible because we must program the first page first, and because programs can only convert 1s to 0s. For 01→00 and 01→01, we only cut off power while programming the second page.

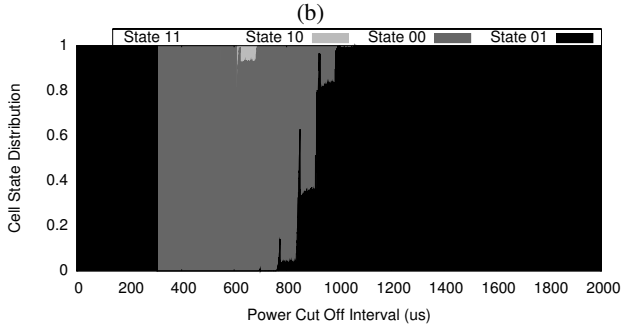
Figure 2 shows the experimental results for B-MLC32-2. For each graph in Figure 2, the x-axis shows the power cut off interval, and the y-axis depicts the distribution of cells for four different states in a block. Figure 2(a) plots the distribution of cell states for the 11→01 transition. The graph shows the shift in state between 0 and 220 μs, but it is not a smooth transition: There are two clear steps with spikes that suggest that some cells temporarily move



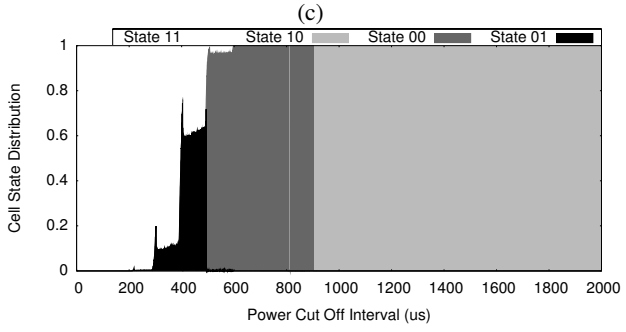
(a)



(b)



(c)



(d)

Figure 2: Cell state breakdown for B-MLC32-2 for (a) 11→01, (b) 01→00, (c) 01→01, (d) 11→10 transitions. The results show that even for seeming no-ops, cells may pass through multiple states

from state 11 to state 01 during programming. The chip reads the cells as state 01 because the second page bits are not programmed yet, but the voltage levels are actually at state 00 instead of state 01 at this point.

Figure 2(b) provides some additional insight into this behavior. It shows the graph for the 01→00 transition. The cell states remain at 01 until 300 μs, when they all instantly become 00. This instantaneous change of cell states indicates, we believe, that the chip switches reference voltages at this point to a new reference that allows the chip to distinguish state 00 from state 01. Since all the cells move to state 00 immediately after the chip applies a new

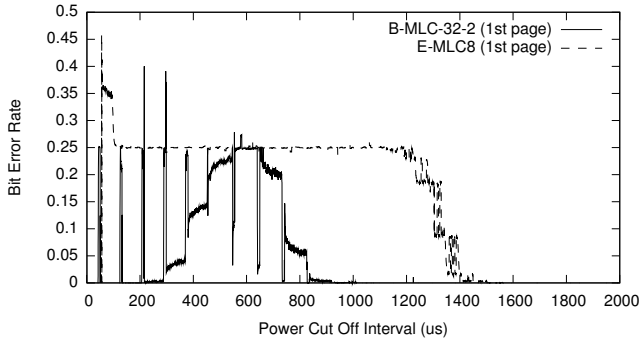


Figure 3: A power failure while programming a second page can corrupt data programmed to the corresponding first page, even if the first page program completed without interruption.

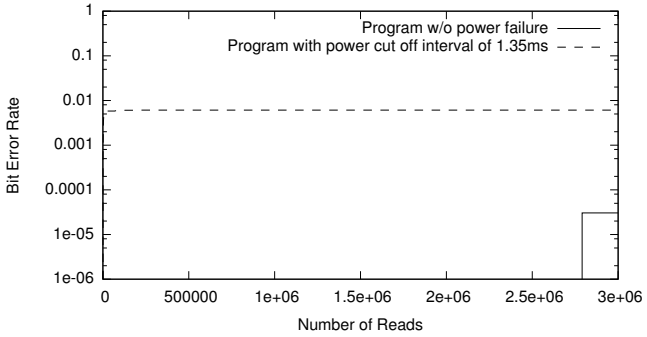


Figure 4: Incompletely programmed pages are more susceptible to read disturb than completely programmed pages.

reference, it appears that the cells were already at a voltage level corresponding to a 00 state after programming the first page was complete.

Figure 2(c) shows the result when we try to perform a 01→01 transition, a seeming no-op. However, if power cuts off between 250 and 1000 μs, a large fraction of the cells will be in state 00 and some may be in state 10.

In Figure 2(d), we make a 11→10 transition. Though we only change the second page bits, the cells move through all possible states during the operation. The chip changes state from 11 to 10 (a shift of one voltage level) during 200 μs to 600 μs. Between 500 μs and 900 μs, it seems to adjust reference voltage to differentiate between states 00 and 01 and result in the abrupt transitions from state 00 to state 10. Then it applies a new reference voltage at 900 μs. The chip can differentiate state 10 from state 00 with the new threshold voltage. This also causes the transition of cell states from 00 to 10.

4.1.2 Retroactive data corruption

The unpredictable effect of power loss during an MLC program operation demonstrated above makes it clear that SSDs must assume that data written during an interrupted program is corrupt. However, the data in Figure 2 also show something more dangerous: Power failure while programming a second page can corrupt data that the chip successfully programmed into a first page. We call this effect *retroactive data corruption*.

Figure 2(d) demonstrates the phenomenon. We expect the program operation to move the cell from 11 to 10, leaving the first page's data untouched. However, we can find cells in any of the four states depending on when power failure occurred.

Figure 3 illustrates this effect in more detail. In this graph, we first program random data to first page bits in B-MLC32-2 and E-MLC8 without power failure. Then we cut off power when we program the corresponding second page bits with random data. The x-axis shows the power cut off intervals for second pages, and the y-axis shows the bit error rates for the first pages. For B-MLC32-2, the bit error rate of the first page reaches 25% with power cut off interval between 200 μs and 900 μs even though the program operation of the first page completed successfully! For E-MLC8, the retroactive data corruption effect is more serious. The bit error rate can reach 50% if the power cut off interval for the second page is between 50 μs and 100 μs .

Flash device datasheets make no mention of this phenomenon, so FTL designers may assume that once a program operation completes, the data will remain intact regardless of any future failures. This assumption is incorrect for MLC devices. Since retroactive data corruption can affect both user data and FTL metadata, it poses a serious threat to SSD reliability.

Besides backup batteries and capacitors, SSDs can take (at least) three steps to mitigate the effects of retroactive data corruption. First, the FTL could program corresponding pairs of first and second pages together and treat them as a logical unit. If programming the second page failed, the FTL would consider them to both have failed. This is not as easy as it sounds, however, since first and second pages are not logically consecutive within the flash block: In most cases the first page at address n is paired with the second page at address $n+6$. Flash device datasheets require programming pages in a block in order because of the flash array organizations. However, our experiment shows that for chips like E-MLC8, programming the first and second pages together does not increase the program error rate.

Second, since the retroactive data corruption never affects the second page, the FTL could protect its metadata by storing it solely in second pages. While it would leave user data exposed to retroactive data corruption, the SSD would, at least, remain operational.

Third, the FTL could adopt a specialized data encoding that would avoid the cell state transitions that can lead to retroactive corruption. For E-MLC8, corruption occurs only when making a 01→01 transition. Sacrificing some bit capacity and applying some simple data coding techniques could prevent that transition from occurring. However, for B-MLC32-2, this scheme does not work since the retroactive data corruption happens in all the cases where the first page bit is 0.

4.1.3 Read disturb sensitivity

Power failure can also affect the data integrity of programmed data by making it more susceptible to other flash failure modes. In this section, we examine the relationship of power failure and read disturb.

Read disturb arises because reading data from a flash array applies weak programming voltages to cells in pages not involved in the read operation. Measurements in [3] shows that it typically takes several million read operations to cause significant errors due to read disturb.

Figure 4 compares the read disturb sensitivity of pages programmed to completion (i.e., no power cut off) and pages programmed with a power cut off interval of 1.35 ms using B-MLC32-2. For that interval, reading the page back reveals no errors.

For both sets of pages, the error rate starts at 0 after programming. For the completely programmed page, errors from read disturb appear after 2.8 million reads. For the partially programmed page, errors appear after just 1000 reads and the error rate rises quickly to 3.1×10^{-3}. It appears that the power failure prevents the program operation from completely tuning the voltage level on some of the cells leaving them susceptible to read disturb.

This effect is potentially dangerous, especially given the very steep increase in error rate. A common approach to dealing with

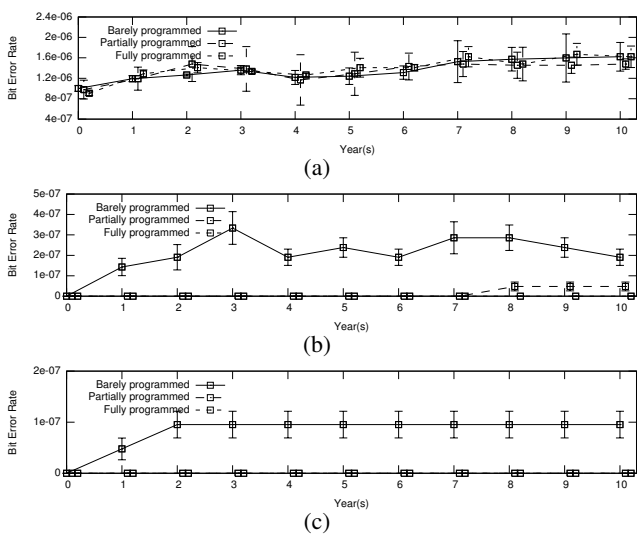


Figure 5: Baking chips to accelerate aging reveals that power failure during program operations reduces the long-term reliability of data stored in flash chips.

read disturb is to copy data to a fresh page when errors begin to appear. However, if the error rate rises too quickly, the data may become uncorrectable before this can occur. The flash memory controller should copy the data programmed under power failure to a fresh page as soon as possible.

4.1.4 Data Retention

Programming data with power failure may also reduce the long-term stability of the data stored in the flash chip. We use a laboratory oven to accelerate the aging effect of flash memory chips up to 10 years. According to the JESD22-A113 standard [5], we bake the chips for 9 hours and 20 minutes at 125°C to achieve the aging of one year. For each chip, we programed 5 blocks for each of the three conditions: (1) Barely programmed: the power cut off interval is as short as possible without resulting in increased bit error rates (for some MLC chips, the programmed error rate is never zero). (2) Fully programmed: the program operation completes without power failure. (3) Partially programmed: the power cut off interval is halfway between barely programmed and fully programmed.

Figure 5 shows the result for (a) B-MLC32-2, (b) E-MLC8, and (c) B-SLC4. The x-axis of the graph is accelerated age in years. The y-axis shows the average bit error rates in a block after aging for each accelerated year. We slightly shift the points horizontally for partially programmed and fully programmed results to make the error bars visible. B-MLC32-2 does not exhibit any relationship between power failure and data retention. However, for E-MLC8, the effect is clear. After 10 years, the error rate for barely programmed data is 1.91×10^{-7} rather than 4.77×10^{-8} for partially programmed or 0 for fully programmed. B-SLC4 shows the robustness common among SLC chips. The chip only shows errors for the barely programmed case.

4.2 Erase and power failure

Erase operations are subject to a different set of reliability requirements than program operations. While it is important that an erase operation reliably write a '1' to every bit in a block, it is equally important that the erase prepare the block properly for the program operations that will follow it. We investigate both aspects of this reliability below.

4.2.1 Erasing bits

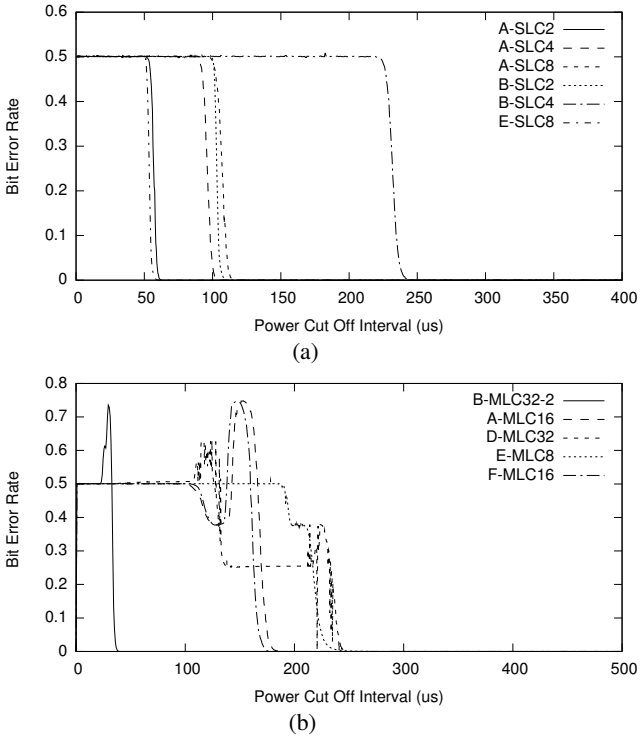


Figure 6: Bit error rates after an interrupted erase operation are well-behaved for SLC devices (a), but MLC behavior is much more complex (b).

Figure 6 presents the bit error rates of erase operations for different power cut off intervals for (a) SLC chips and (b) MLC chips. For each test, we initially programmed the block with random data.

Behavior is similar among the SLC chips that we tested. The bit error rate stays at 50% (since the block contains random data) for between 50 and 240 μs, after which all the cells become erased in less than 10μs. However, chips do not report that the command has completed until 400 μs– 2 ms.

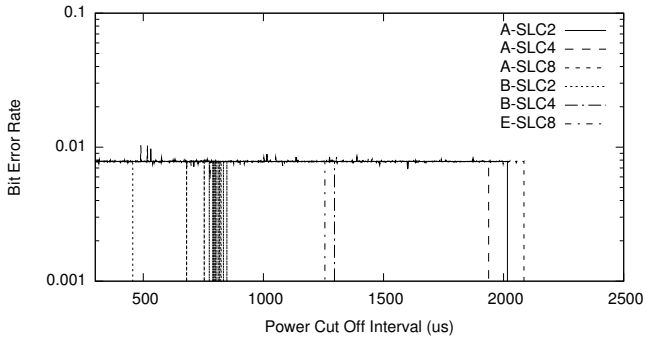
For MLC chips, the bit error rate is, once again, non-monotone. For all chips except E-MLC8, it reaches as high as 75%. The timing of MLC chips also varies among different models: It takes between 50 μs – 475 μs for every cell to become 1. However, these chips report that erase command completes after 2 – 4 ms.

4.2.2 Programming blocks after an erase failure

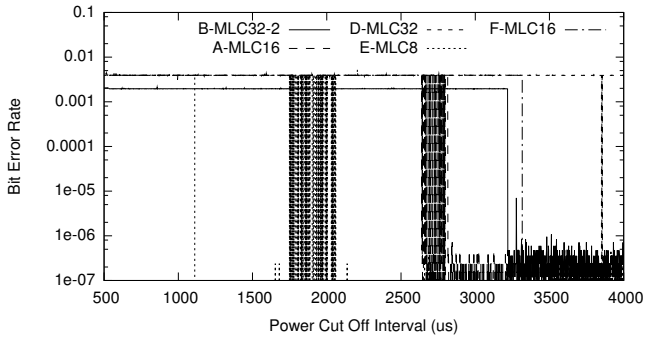
In previous experiments, we found that erases appear to be complete long before the chip reports that they are finished. Based on discussions with flash manufacturers, we suspect that the chip spends this time fine-tuning the voltage levels in the cells. Presumably, this step is important (or the manufacturers would save time and skip it), so it is important to understand what impact cutting it short will have.

To measure this effect, we cut power during erase operations, performed a complete program operation, and then read back the data to measure the bit error rate.

Figure 7 shows the results for (a) SLC and (b) MLC chips. The x-axis measures the power cut off interval of the previous erase operation and the y-axis depicts the bit error rates of the later programming operations. We start the experiment at 300 μs for SLC and 500 μs for MLC, since by this time, the erase appears to be complete. In both cases, interrupting an erase operation reduces reliability for future program operations to the block. For SLC and most MLC chips, the error rate rises from 0 to between 0.4% and 0.9%. For B-MLC32-2, the program error rate is never zero, and the bit error rate rises from 1.2×10^{-7} to 0.2%. For E-MLC8, the



(a)



(b)

Figure 7: Bit error rates after an interrupted erase operation are well-behaved for SLC devices (a), but MLC behavior is much more complex (b)

program error rate varies between 0.4% and 0% for power cut off intervals between 1747 and 2062 μs. The program error rate of A-MLC16 also bounces for power cut off interval between 2640 μs and 2814 μs. These frequent variations cause the two vertical bands in the graph.

5. RELATED WORK

Flash manufactures provide limited information about many aspects of their chips, including their behavior under power loss. Our work is similar in spirit to Grupp et al. [3] in that we empirically quantify flash behavior in order to better understand the opportunities and design challenges that it presents. In addition to chip level performance, Boboila et. al [1] also explored device-level characteristics including timing, endurance, and FTL designs. However, neither of the above works focus on the power failure behavior of flash memory chips.

Many high-end SSDs have backup batteries or capacitors to ensure that operations complete even if power fails. Our results argue that these systems should provide power until the chip signals that the operation is finished rather than until the data appears to be correct. Low-end SSDs and embedded systems, however, often do not contain backup power sources due to cost or space constraints, and these systems must be extremely careful to prevent data loss and/or reduced reliability after a power failure.

Existing work on recovery from power failure aims to restore or repair flash file systems using logs and other techniques [16, 4, 12, 10] or page-level atomic writes [7, 2]. Numonyx [11] also provides guidelines to repeat interrupted operations after power failure. These designs may work for SLC chips, but the retroactive data corruption we observed for MLC chips suggests that they will be less effective there.

Some commercial systems avoid retroactive corruption by treating a block as the basic unit of atomic writes [9, 13, 14]. This approach is inefficient for small writes since it requires re-writing at least one entire block for each write operation. We described several alternative solutions.

System software and embedded applications are critical in dealing with reliability issues like power loss. Kim et al. [8] designed a software framework to mimic faults, including power failure, in flash memory. This work (and the results in [3]) demonstrates that flash has many non-intuitive error modes, so fault-injection frameworks require input from real hardware measurements to ensure the faults they inject are representative of what can occur in real systems.

6. CONCLUSION

The flash memory devices we studied in this work demonstrated unexpected behavior when power failure occurs. The error rates do not always decrease as the operation proceeds, and power failure can corrupt the data from operations that completed successfully. We also found that relying on blocks that have been programmed or erased during a power failure is unreliable, even if the data appears to be intact.

7. REFERENCES

[1] S. Boboila and P. Desnoyers. Write endurance in flash drives: measurements and analysis. In *FAST '10: Proceedings of the 8th USENIX conference on File and storage technologies*, pages 9–9, Berkeley, CA, USA, 2010. USENIX Association.

[2] T.-S. Chung, M. Lee, Y. Ryu, and K. Lee. Porce: An efficient power off recovery scheme for flash memory. *Journal of Systems Architecture*, 54(10):935 – 943, 2008.

[3] L. Grupp, A. Caulfield, J. Coburn, S. Swanson, E. Yaakobi, P. Siegel, and J. Wolf. Characterizing flash memory: Anomalies, observations, and applications. In *MICRO-42: 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 24 –33, 12 2009.

[4] A. Gupta, Y. Kim, and B. Urgaonkar. DFTL: a flash translation layer employing demand-based selective caching of page-level address mappings. In *ASPLOS '09: In Proceeding of the 14th international conference on Architectural support for programming languages and operating systems*, pages 229–240, 2009.

[5] JEDEC. Preconditioning of Plastic Surface Mount Devices Prior to Reliability Testing. http://www.jedec.org/sites/default/files/docs/22a113F.pdf.

[6] T.-S. Jung, Y.-J. Choi, K.-D. Suh, B.-H. Suh, J.-K. Kim, Y.-H. Lim, Y.-N. Koh, J.-W. Park, K.-J. Lee, J.-H. Park, K.-T. Park, J.-R. Kim, J.-H. Yi, and H.-K. Lim. A 117-mm2 3.3-v only 128-mb multilevel nand flash memory for mass storage applications. *IEEE Journal of Solid-State Circuits*, 31(11):1575 –1583, Nov. 1996.

[7] J. Kim, J. M. Kim, S. H. Noh, S. L. Min, and Y. Cho. A space efficient flash translation layer for compactflash systems. *IEEE Transactions on Consumer Electronics*, 48:366–375, 2002.

[8] S.-K. Kim, J. Choi, D. Lee, S. H. Noh, and S. L. Min. Virtual framework for testing the reliability of system software on embedded systems. In *SAC '07: Proceedings of the 2007 ACM symposium on Applied computing*, pages 1192–1196, New York, NY, USA, 2007. ACM.

[9] K. Y. Lee, H. Kim, K.-G. Woo, Y. D. Chung, and M. H. Kim. Design and implementation of mlc nand flash-based dbms for mobile devices. *Journal of Systems and Software*, 82(9):1447–1458, 2009.

[10] P. March. Power Loss Recovery (PLR) for cell phones using NAND Flash memory. http://www.numonyx.com/en-US/ResourceCenter/SoftwareArticles/Pages/PLRforNAND.aspx.

[11] Numonyx. How to operate Power Loss Recovery for the Numonyx 65nm Flash Memory Devices. www.numonyx.com/Documents/Applications_Operation.pdf.

[12] C. Park, P. Talawar, D. Won, M. Jung, J. Im, S. Kim, and Y. Choi. A High Performance Controller for NAND Flash-based Solid State Disk (NSSD). In *NVSMW '06: Non-Volatile Semiconductor Memory Workshop, 2006.*, pages 17 –20, feb. 2006.

[13] S. Park, J. H. Yu, and S. Y. Ohm. Atomic write FTL for robust flash file system. In *ISCE '05: Proceedings of the Ninth International Symposium on Consumer Electronics, 2005.*, pages 155 – 160, June 2005.

[14] F. M. Systems. Power failure prevention, recovery, and test. http://www.fortasa.com/Attachments/040_Power Failure Corruption Prevention.pdf.

[15] K. Takeuchi, T. Tanaka, and T. Tanzawa. A multipage cell architecture for high-speed programming multilevel NAND flash memories. *IEEE Journal of Solid-State Circuits*, 33(8):1228 –1238, Aug. 1998.

[16] D. Woodhouse. JFFS2: The Journalling Flash File System, version 2. http://sources.redhat.com/jffs2/.