SSSI TECH NOTES

# How Controllers Maximize SSD Life

January 2013

SNIA
Solid State Storage Initiative

by SNIA SSSI Member:

**Jim Handy**

Objective Analysis

*"The SSD Guy"*

www.snia.org

## About the Solid State Storage Initiative

The SNIA Solid State Storage Initiative (SSSI) fosters the growth and success of the market for solid state storage in both enterprise and client environments. Members of the SSSI work together to promote the development of technical standards and tools, educate the IT communities about solid state storage, perform market outreach that highlights the virtues of solid state storage, and collaborate with other industry associations on solid state storage technical work.

SSSI member companies come from a wide variety of segments in the SSD industry **www.snia.org/forums/sssi/about/members**.

# How Controllers Maximize SSD Life

by SNIA SSSI Member:

**Jim Handy**
"*The SSD Guy*", Objective Analysis

**Table of Contents**

# Introduction

This booklet contains a collection of posts from Jim Handy's SSD Guy blog **www.TheSSDGuy.com** which explores the various techniques designers use to increase SSD life.

# How Controllers Maximize SSD Life

How do controllers maximize the life of an SSD? After all, MLC flash has a lifetime of only 10,000 erase/write cycles or fewer and that is a very small number compared to the write traffic an SSD is expected to see in a high-workload environment, especially in the enterprise. Still, MLC is becoming the norm in the enterprise.

## How Do They Do That?

This is where SSD architects really earn their pay. There are eight basic techniques that The SSD Guy knows of to extend SSD life beyond the wear that the flash alone would provide:

- Wear Leveling
- External Data Buffering
- Improved ECC
- Other Error Management
- Reduced Write Amplification
- Over Provisioning
- Feedback on Block Wear
- Internal NAND Management

What is important to consider is that the right mix of these techniques allows an SSD to have significantly better wear than the internal NAND's specifications would support.

# Better Wear Leveling

First we will explore how the right wear leveling algorithm can help a controller maximize the life of an SSD.

Wear leveling is a fact of life with NAND flash – blocks start to suffer bit failures after a certain number of erase/write cycles (usually specified from the thousands to the hundreds of thousands) and it is only natural that software will attempt to over-write some blocks more than others. In order to prevent this from causing failures, all of today's SSD, USB flash drive, and flash card controllers incorporate some sort of wear leveling.

This is a simple re-mapping of the contents of the flash chips. A more graphical explanation is given in any of Objective Analysis' SSD reports.

When the software attempts to write to a certain block, it is automatically forced to write somewhere else. The next write is likely to be mapped somewhere yet again. This is done in a way that is completely invisible to the rest of the world, and results in all of the blocks in the flash chip receiving roughly the same amount of wear.

Wear leveling algorithms are not all the same, though, and some work better than do others.

There are two fundamental categories of wear leveling: Static and Dynamic. With dynamic wear leveling only two kinds of blocks participate in the wear leveling algorithm: those that undergo change, and those that are unused. Blocks of data that don't change ("Static" blocks) are left alone by a dynamic write leveling algorithm.

Static wear leveling manages write leveling across all of the flash in the system – the changed blocks, the unused blocks, and the static blocks. This means that a static block that contains valid data but that is never being written to will still be moved around within the flash in order to allow all blocks to receive the same amount of wear. It's easy to understand why this would help, since those blocks would otherwise be unavailable to the wear leveling algorithm. On the other hand, if you're trying to minimize write/erase cycles, it's unintuitive that this approach is the better of the two.

Underneath these two fundamental wear leveling algorithms lie several other strategies that The SSD Guy won't go into here. These determine how much data will be moved, where it will be moved, and how data fragments can be consolidated to make one full block out of a number of partial blocks.

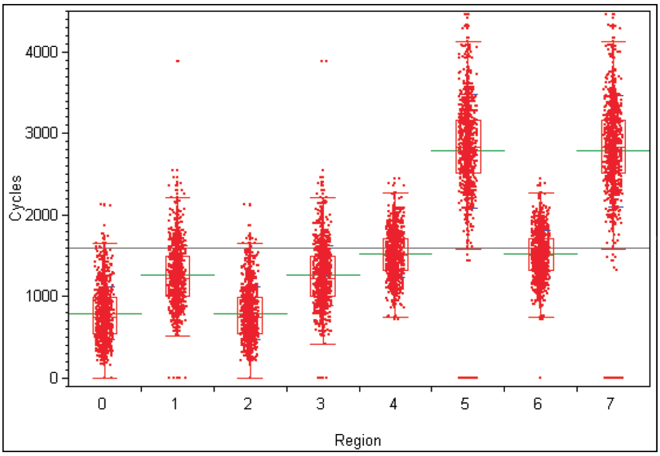
Controller designers have to walk a tightrope when it comes to wear algorithms since a very aggressive wear leveling algorithm may create an excessive number of unnecessary additional writes. Meanwhile, the wear algorithm has to interact well with other operations within the drive (like garbage collection) or it might also create undue block wear.

This is all very tricky!

The selection of these algorithms determines not only how much wear each block will receive, but also how quickly the SSD will perform in demanding applications. Each SSD architect has a unique philosophy about how these algorithms should be managed. The best algorithm depends on a number of variables including NAND type, workload, speed requirements, and so on.

Intel has shared some graphics that help illustrate how wear leveling algorithms differ. The first is a chart of an SSD with a very basic wear leveling algorithm called "Regioned Leveling." The X-axis represents different flash regions. The Y-axis indicates the number of erase/write cycles each has undergone. A red dot represents the activity in and individual block. Clearly some blocks, even blocks within the same region, undergo significantly more write activity than do others, and these blocks should be the first to fail.

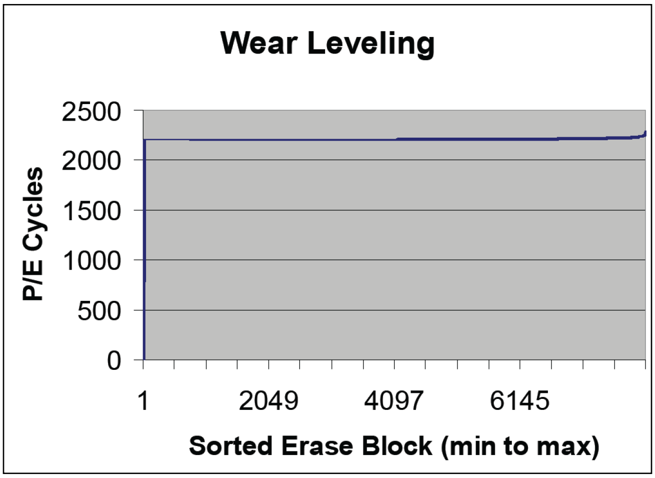
**Figure 1 - Erratic wear of a basic wear leveling algorithm.**



Source: Intel Corp., Used with permission.

The second graphic contains similar data for an Intel SSD, but in this chart the data has been sorted from lowest to highest because otherwise the difference between blocks might not clearly show up in the chart. The Intel SSD used in this example uses a much more balanced wear leveling algorithm that will wear each block at essentially the same rate. According to Intel, the first SSD exhibited some cases in which peak wear was three times the average. The second had only a 4% difference between the blocks with the maximum and minimum wear.

**Figure 2 - Wear leveling in an Intel SSD**



Source: Intel Corp., Used with permission.

What impact does this have on SSD longevity? Well, if the wear leveling algorithm can both minimize erase/write cycles and truly even out wear across the entire SSD, then it will naturally coax the longest life out of the blocks before any of them starts to wear.

## External Data Buffering

Since NAND flash is weakened by erase/write cycles then it would make sense to try to reduce those cycles to prolong the life of an SSD right? That's what external data buffers are designed to do.

There are many ways to use RAM (either a RAM internal to the SSD controller chip or a discrete DRAM chip on the SSD's printed circuit card) to stage data in a way that will reduce erase/write cycles.

One is to perform a function called "Write Coalescing." This involves gathering several short writes to adjacent SSD sectors to turn them into a single long write from the buffer into the NAND flash. One large write is less taxing to the chip than are several small writes. It's also a lot faster.

Here's an example: In an SSD without a write buffer a number of short writes to the same general area in the drive may not all occur at the same time. An early write is likely to already be committed to the flash before a later one is received by the SSD. When the later write is received, the flash block that contains both sectors may need to be reassigned to another block in order to perform the operation, leading to an additional erase/write cycle.

Write coalescing is not the only way a RAM buffer can be used to reduce write traffic to the flash in an SSD. Another approach involves buffering several successive writes to the same sector, without actually writing to the flash until several such writes have been performed. Although the system may believe that it is repeatedly overwriting the same NAND sector, in truth the overwritten sector is in RAM until that RAM is needed for some other task. Several hundred writes to a single sector may turn into a single write to NAND. This means that in some cases the endurance of the NAND may be increased by a couple of orders of magnitude, which is a pretty good return for a couple of dollars' worth of RAM.

RAM is also used to assure that writes occur in full page lengths, since this is the away that NAND most efficiently performs a write. By matching the length of a NAND write to the chip's page length the buffer helps to reduce write traffic. Different NAND chips have different page lengths (generally 1-4kB) so the controller has to be informed of the page length of the exact NAND chips used in the SSD.

Any of these approaches creates a risk that data in RAM will be lost during a power failure. The work-around for this is to provide internal energy storage that can keep the SSD alive long enough to write the RAM's contents into the NAND for an orderly shut-down. The power for this process is commonly provided one of three ways – it can be from a:

1. Battery
2. Super Capacitor
3. Bank of Tantalum Capacitors

We won't join the debate about which of these is best.

Still, it's clear that adding a RAM as temporary storage can do a lot to lengthen flash lifetime well beyond the limit of its endurance specifications.

## Improved ECC

Error correction codecs (ECC) can have a very big impact on the longevity of an SSD, although few understand how such a standard item can make much difference to an SSD's life. The SSD Guy will try to explain it in relatively simple terms here.

All NAND flash requires ECC to correct random bit errors ("soft" errors.) This is because the inside of a NAND chip is very noisy and the signal levels of bits passed through a NAND string are very weak. One of the ways that NAND has been able to become the cheapest of all memories is by requiring error correction external to the chip.

This same error correction also helps to correct bit errors due to wear. Wear can cause bits to become stuck in one state or the other (a "hard" error), and it can increase the frequency of soft errors.

Although it is not widely understood, flash endurance is the measure of the number of erase/write cycles a flash block can withstand before hard errors might occur. Most often these failures are individual bit failures – it is rare for the entire block to fail. With a high enough number of erase/writes the soft error rate increases as well due to a number of mechanisms that I won't go into here. If ECC can be used to correct both these hard errors and an increase in soft errors then it can help lengthen the life of a block beyond its specified endurance.

Here's an example: Let's say that an unworn NAND chip has enough soft errors to require 8 bits of ECC – that is, every page read could have as many as 8 bits that have been randomly corrupted (usually from on-chip noise.) If the

ECC that is used with this chip can correct 12 bit errors then a page would have to have 8 soft errors plus another 5 wear-related errors to go beyond the ECC's capability to correct the data.

Now the first of those 5 failures is guaranteed by the flash manufacturers to occur sometime after the endurance specification: No bits will fail until there have been 10,000 (or 5,000, or 3,000…) erase/write cycles. Specifications are not sophisticated enough to predict when the next bit will fail, but there may be several thousand more cycles before that occurs. This implies that it will take significantly more than the specified endurance before a page becomes so corrupt that it needs to be decommissioned. This means that the error-corrected endurance of the block could be many times the specified endurance, depending on the number of excess errors the ECC is designed to correct.

This all comes at a price. More sophisticated ECC requires more processing power in the controller and may be slower than less sophisticated algorithms. Also, the number of errors that can be corrected can depend upon how large a segment of memory is being corrected. A controller with elaborate ECC capabilities is likely to use more compute resources and more internal RAM than would one with simpler ECC. These enhancements will make the controller more expensive.

ECC algorithms are their own special other-worldly state of mathematics, and although The SSD Guy is pretty comfortable with math, but I don't even try to understand the difference between the more basic Reed-Solomon coding and LDPC. When someone talks to me about the Shannon Limit (the maximum number of bits that can be corrected) my eyes glaze over. Suffice it to say that I am awestruck by the intelligence of the folks who have mastered ECC, and their ability to extract more life out of a flash block than any mere mortal would think possible.

Just remember that more bits of error correction leads to a longer usable life for a flash block before it needs to be decommissioned.

## Other Error Management

There are more advanced means than simple error correction to help remove bit errors in NAND flash and those will be the subject of this section. The general term for this approach is "DSP" although it seems to have very little to do with the kind of DSP algorithm used to perform filtering or build modem chips.

While ECC corrects errors without knowing how they got there, DSP helps to correct any of the more predictable errors that are caused by internal error mechanisms that are inherent to the design of the chip. A prime example of such an error would be adjacent cell disturb.

Here's a brief explanation of adjacent cell disturb: Since each bit cell in a flash chip is like one of an array of tiny capacitors all manufactured on the same die, there can be cross coupling between adjacent cells. An adjacent cell may bleed charge off of a neighboring cell. During reads and writes the energy passing through the adjacent cell may push a neighboring cell's bit just past its threshold – during a read the current drawn off the adjacent cell may draw some charge off a neighboring cell, and during a write the high fields in the adjacent cell may increase the charge of its neighbor.

The DSP engine knows which bits in a chip are adjacent to which other bits, and actually changes the data written into the flash into "symbols" (a term from communications technology). The symbols are chosen to make the data less sensitive to errors stemming from the effects of adjacent cell disturb.

There is a science surrounding the selection of the right symbols, since they impact other aspects of the SSD including write speed.

When reading the flash the symbols are re-mapped into data, a process that corrects predictable errors before they are processed through the ECC algorithm.

DSP corrects the more predictable errors leaving the less predictable errors for the ECC algorithm to clean up.

Since cleaner data is passed to ECC for correction, DSP effectively adds more bits of error correction to the ECC. As we saw in the ECC post, adding bits of error correction can extend the endurance of the flash chip, so in an indirect way, DSP not only corrects errors, but it also extends the life of the flash in an SSD.

DSP is not all that widely used today, but a growing number of future SSD controllers are likely to include this approach.

## Reduced Write Amplification

Write amplification plays a critical role in maximizing an SSD's usable life. The lower the write amplification, the longer the SSD will last. SSD architects pay special attention to this aspect of controller design.

Unlike the other factors described in this series this is not a technique that extends flash life beyond the 10,000 erase/write cycles that one would normally expect to result in a failure, but it is very important to SSD longevity.

An SSD that is trying to manage its free blocks will need, at times, to move data from two sparsely-populated blocks to another block so that those sparsely-populated blocks can be erased. Since the host system didn't ask the SSD to perform this write, it results in a write that is over and above what the host system requested, so the SSD undergoes more writes than the host system intended; that's write amplification.

If the number of NAND writes within the SSD matched the number of host system write requests then the Write Amplification (or Write Amplification Factor – WAF) would be 1.0. If the average number of writes to flash were twice the number of host system write requests then the Write Amplification would be 2.0.

If you were to write a piece of system software that measured SSD writes, this measurement would fall short of the actual number of times the flash chips were being written to because of Write Amplification.

There are many ways to reduce Write Amplification. Some SSDs write data to the SSD in a log structure – the data is written into sequential NAND addresses and housekeeping is performed during idle times. This does reduce write amplification, but data management gets complicated as the SSD fills.

SandForce's approach is to compress the data going into the SSD. If the data is compressed to 1/2 its normal size, then half the system writes actually make it into the SSD's NAND chips. This allows SandForce to boast that its write amplification is less than 1.0, a figure that is not intuitive unless you understand the approach the company uses.

In brief, an SSD that has been designed to reduce write amplification will reduce the number of times the NAND flash sees write cycles, thus taking the best advantage of the limited number of erase/writes that its NAND flash chips can withstand before experiencing bit errors.

## Over Provisioning

Over provisioning is one of the most common ways that SSD designers can help assure that an SSD has a longer life than the flash's endurance rating would support. If an SSD contains more flash than is presented at its interface, the controller can manage wear across a larger number of blocks while at the same time accelerating disk performance by moving slow operations like block erases out of the way of the SSD's key functions.

Many people like to compare wear leveling to rotating a car's tires. In this vein, think of over provisioning as having a bunch of spare tires (rather than just one) in the trunk of your car. The more tires you have, the farther you can drive before having to buy a new set of tires, although at any one time you are never using more than four tires. On the other hand, this pushes up the cost of a tire change a good bit.

Over provisioning also helps reduce write amplification. When an SSD gets full the garbage collection algorithm works to consolidate partial blocks to provide more free space. An SSD with a lot of spare blocks doesn't need to go through this process as frequently as an SSD with few spare blocks.

(A related, but somewhat different way that blocks are freed is through the Trim command. The Trim command allows the operating system to tell the SSD which sectors are no longer useful. This allows the SSD to erase unused blocks well in advance of their use, reducing the likelihood that the SSD will fill up.)

How much does over provisioning help? Based on inputs from a few companies The SSD Guy concludes that, for standard MLC NAND flash, 45% over provisioning will give you about twice the drive life of 20% over provisioning, and 75% over provisioning will extend the disk's life to three times. A full 100% over provisioning won't get you quite as far as four times the drive life, but it comes close. Increases in over provisioning show a trend of diminishing returns approaching zero at around 250% over provisioning (and about 6 times the life of that 20% over provisioned SSD).

Many contend that 7% over provisioning is a standard for client SSDs. This level would provide about 1/3rd of the lifetime of a 20% over provisioned SSD.

Something very weird happens with compression, which is used in SandForce controllers. These controllers compress the data as it is being written into the NAND flash. This means that the amount of over provisioning is a function of how compressible the data is. Highly-compressible data will leave a lot of flash unused for data storage, so the SSD will have a very large amount of over provisioning, extending its wear significantly. Incompressible data (usually data that has already been compressed, like MP3 files, JPEG photos, or video files) will consume most of the flash in the SSD leaving very little for over provisioning. SSDs storing incompressible data will wear out after fewer disk writes. Quite fortunately, applications with high write workloads do not usually manage these file types.

A note about the terminology: Some folks spell this as a single word – Overprovisioning – and even The SSD Guy is inconsistent about which way to write it. Since it's a term that began its days decades ago with RAID you would expect some standard to already be in place, but this seems not to be the case. I suggest picking one that feels right to you and overlooking the inconsistency across the industry.

## Feedback on Block Wear

One way that SSD controllers maximize the life of an SSD is to use feedback on the life of flash blocks to determine how wear has impacted them. Although this used to be very uncommon, it is now being incorporated into a number of controllers.

Here's what this is all about: Everybody knows that endurance specifications tell how much life there is in a block, right? For SLC it is typically 100,000 erase/write cycles, and for MLC it can be as high as 10,000 cycles (for older processes) but goes down to 5,000 or even 3,000 for newer processes. TLC endurance can be in the hundreds of cycles. Now the question is: "What happens after that?"

In most cases individual bits start to stick after that. How long after that, though, is not fully specified. In a SNIA white paper that The SSD Guy helped to write (*NAND Flash Storage for the Enterprise - an In-Depth Look at Reliability*), Fusion-io shared some chip characterization the company performed that showed SLC flash lasting sometimes over 4 million cycles before bits started to stick. (See Figure 2 of this white paper, which shows negligible bit errors below 2 million cycles for SLC NAND.)

If NAND can be programmed and erased so very many times, then why don't chip makers specify the greater number? Wouldn't this give them a competitive advantage? In fact, for the vast majority of their market – consumer electronics – a better wear specification would not help sell flash chips, and it would be very costly to specify it accurately. Endurance tests are very time consuming and unfortunately they destroy the chip they are testing. Add to this the fact that endurance can vary by chips within a single wafer, and the result is that you can't perform lot characterization efficiently, and even if you could, it would create an inventory situation as the chip maker delayed shipping a production batch of chips until the wear testing had been completed.

The answer is to set the bar low enough that every part ever made will pass. In the case of SLC flash this bar is set at 100,000 cycles. For MLC it is set much lower, and for TLC it can be an order of magnitude lower than for MLC.

So, how can this bit of information be used to extend SSD life?

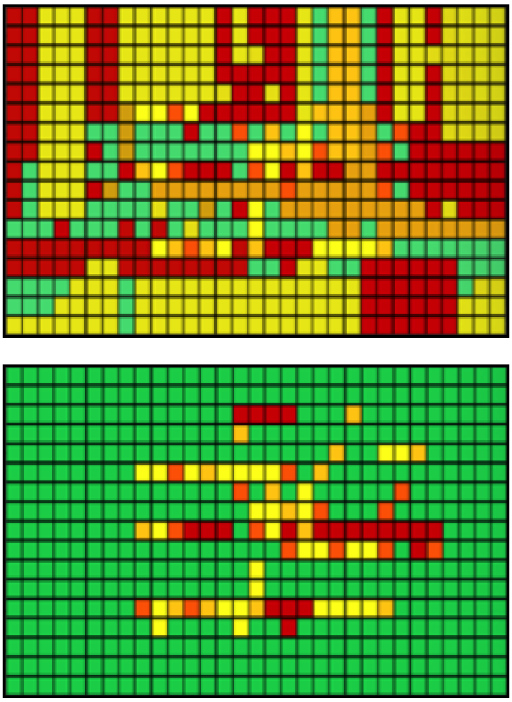
Let's say that your controller can figure out how many erase/write cycles every flash block in the SSD can withstand. Then it can manage the wear leveling algorithm to put more wear onto the hardy blocks and less onto the weaker blocks. More importantly, though, the controller will not simply give up and say: "100,000 erase writes on this block – time to shut it off!" Instead it will take the block right to the point where the bit errors are nearly the maximum number the error correction algorithm can correct.

Naturally, this will extend the life of each NAND block to its absolute limit, most likely multiplying the life of the SSD to many times that of an SSD without this feedback.

It's not a complicated idea, but one that didn't occur to early SSD designers. It's the kind of simple, yet highly-effective approach that I would expect to come into universal use over time

Figure 3 illustrates the results when one SSD maker – SMART Storage Systems – combines this technique with some of the other approaches mentioned in this brochure to extend the life of their SSDs' NAND chips.

**Figure 3. NAND flash wear based on common algorithms (top) and on more sophisticated algorithms based on sensing actual flash wear.**



Images courtesy of SMART Storage Systems. Used with permission.

This represents NAND flash blocks after five years of heavy write loads. The red blocks have failed and are no longer used by the SSD. Yellow blocks are close to failure. Green blocks have plenty of remaining life. Clearly the lower model will give far longer SSD life than the upper model, which is based on an assumption of 10,000 erase/write cycles.

## Internal NAND Management

Given that you have used all those other forms of improving SSD wear that we have discussed so far, but you still don't find that this is enough, what do you do next? Well a few SSD controllers go one step further and manage some of the inner workings of the NAND flash chip itself.

If that sounds like a significant undertaking to you, then you clearly understand why so very few controllers take this approach. The information used to perform this function is not generally available – it takes a special relationship with the NAND flash supplier – and you can't develop this relationship unless the NAND supplier is certain that you won't go sharing its secrets with its competition. This means that the SSD controller is dedicated to a single brand of NAND, and it means that the SSD maker can't shop around among NAND suppliers for the best price. Furthermore, the NAND supplier won't share this information unless it believes that there is some compelling reason to work the SSD manufacturer. Since there are hundreds of SSD makers it's really difficult to get these companies to pay attention to you!

The SSD manufacturers that have this kind of relationship with their flash suppliers are very rare and very special.

Let's say that you're an SSD maker that's in the happy position where a NAND supplier is willing to disclose such information to you. What will you learn? You will be told how to manage a lot of variables that can be controlled by changing internal settings on the NAND flash, and of parameters that influence bit errors:

- Thresholds between levels in MLC
- Programming times and algorithms
- Physical placement relationships between adjacent cells
- Means of de-trapping electrons trapped in the tunnel oxide

There are more than these, but these are the ones The SSD Guy knows about. These are the kinds of things that can be controlled if you understand how to access the special internal test modes that are implanted within every flash chip sold today.

To give you an idea of how this works, let's look at a single parameter: MLC level sensing.

When an MLC flash is programmed it places two bits on a cell by charging the cell to one of four levels:

- Uncharged
- 1/3 charged
- 2/3 charged
- Fully charged

Certain inevitable issues can cause these levels to veer off-center. Trapped electrons in the tunnel oxide can make the floating gate appear to hold a larger charge than the programming algorithm actually placed on it. Thermal or other influences can drain electrons off the cell to reduce the charge. Adjacent cell reads and writes may add or remove electrons from the floating gate.

A controller that understands these phenomena can adjust the sensing thresholds within the NAND chip to help reduce the chance that a bit will be misinterpreted. If a cell is supposed to be 2/3 charged, but it looks like it's 1/2 charged, will it be read as 1/3 charged or as 2/3 charged? If the thresholds are managed by the SSD controller to compensate for other factors the controller can reduce the impact of this source of bit errors.

Clearly this is a pretty extreme way to squeeze the last bit of life out of NAND flash. It is used to push the technology far beyond its specified wear, but the companies that use it are very successful in achieving flash life well beyond that of controllers that don't use this approach.

## The Solid State Storage Initiative

### About SNIA

The Storage Networking Industry Association (SNIA) is a not-for-profit global organization made up of some 400-member companies and 7,000 individuals spanning virtually the entire storage industry. SNIA's mission is to lead the storage industry worldwide in developing and promoting standards, technologies, and educational services to empower organizations in the management of information. To this end, SNIA is uniquely committed to delivering standards, education, and services that will propel open storage networking solutions into the broader market. For additional information, visit the SNIA web site at **www.snia.org**.

SSSI TECH NOTES

## About this Book

A NAND Flash cell has a well-understood lifespan.

Learn about the techniques used by SSD controllers to maximize SSD life.

Wear leveling is important, but there's much, much more involved.

### About the Author

Jim Handy, Director at Objective Analysis is the author of "*The SSD Guy*" blog and is an active member of the SSSI.

**www.objective-analysis.com**
**www.thessdguy.com**

www.snia.org
www.snia.org/forums/sssi
www.sniasssiblog.org

**Storage Networking Industry Association**
425 Market Street, Suite 1020
San Francisco, CA 94105
Phone: 415-402-0006
Fax: 415-402-0009

SNIA
Solid State Storage Initiative