SNIA
Solid State Storage Initiative

NAND Flash Solid State Storage for the Enterprise

An In-depth Look at Reliability

April 2009

SSSI Members and Authors:
Jonathan Thatcher, Fusion-io
Tom Coughlin, Coughlin Associates
Jim Handy, Objective-Analysis
Neal Ekker, Texas Memory Systems

The Solid State Storage Initiative

Recognizing the dramatically escalating interest in solid state storage, and thus an increased need for accurate technology information, education, and standards development, the Storage Networking Industry Association (SNIA) formed a new body in 2008, the Solid State Storage Initiative (SSSI). The SSSI immediately pursued the formation of the Solid State Storage Technical Work Group (SSS TWG) which works hand-in-hand with the SSSI. Many educational articles, presentations, and tutorials are being developed and made available to the public by the SSSI, with support from the SSS TWG, to fulfill its mission of fostering the growth and success of solid state storage in both consumer and commercial environments. This paper is a collaboration of the members of the SSSI. It is being made available to the public to provide information needed to understand this new solid state storage technology.

No matter how the future of solid state storage plays out, the SNIA and its SSS Initiative and TWG will play an important role. The ultimate goal is to further the interests of solid state storage through education, standards development, and overall promotion of appropriate SSS deployment. The group will be a resource to users and the solid state storage community in defining, testing, and refining the rapidly developing solid state storage market. For more information about the SSSI, visit www.snia.org/sssi

Introduction

NAND Flash-based solid state storage (SSS) solutions, as they exist today, offer unparalleled performance combined with a level of data integrity and availability for mission-critical data that matches and potentially exceeds storage solutions based on mechanical, magnetic drives. Long associated with consumer electronics, NAND Flash has become a viable storage medium for commercial and governmental information systems, often referred to collectively as enterprise applications.

It is well established by now that SSS solutions outperform storage systems based on hard disk drives (HDDs), even when HDDs are combined into large, multi-disk, striped arrays, short-stroked, or managed in other ways to increase performance. Less well known, however, due to some lingering misperceptions about NAND Flash as a storage medium, is the fact that NAND Flash-based SSS doesn't just outperform its mechanical counterpart—it also significantly improves on data integrity (confidence that what you put in for storage is exactly what you get out) and availability (confidence that your data can be retrieved whenever you want it).

Past comparisons between SSS and mechanical drives have frequently focused on single devices—how one SSS drive stacks up against one mechanical drive. However, enterprise computing environments rarely rely on isolated direct attached storage (DAS). The value of SSS becomes more apparent as used in real world data centers: as part of network attached storage (NAS) systems or storage area networks (SAN), where SSS can simultaneously increase the performance while greatly reducing the overall complexity and resource costs of data centers.

In and of itself, NAND Flash does have some performance peculiarities and some reliability challenges, but this paper will show that when combined with good management, NAND Flash-based SSS is a high-performing, highly reliable storage solution.

This paper will discuss:

- background information about the evolution of NAND Flash,
- the challenges of using NAND Flash as a storage medium, and in particular, those that have the potential to affect data integrity and availability, and
- ways that well-designed controllers and management software manage NAND Flash to simultaneously achieve both performance and reliability.

Evolution of Flash

History

Flash memory is a 1980s invention that offered a revolutionary property to the integrated circuit community: it could be both programmed and erased electronically and retain its state without power being applied just like the Electronically Programmable Read Only Memory (EPROM) and Electronically Erasable/Programmable Read Only Memory (EEPROM) that preceded it. RAM chips have been available since the 1960s allowing electronic reading and rewriting, but power has to be constantly applied for the RAM to maintain its state. In the early 1970s EPROM chips arrived, allowing manufacturers to electronically program chips that would retain their state without power, but after programming they could only be erased by removing the chip from the system and exposing it to ultraviolet light for an hour. In the early 1980s EEPROM was invented, becoming the first chip that could be both programmed and erased electronically, without being removed from the system, and maintain the programmed value without power.

EEPROM was primarily used to store programs for embedded processors and thus only needed to change when programmers fixed bugs or added new features. This happened fairly infrequently, so EEPROM was ideally suited for this purpose. Programming the EEPROM took several seconds to several minutes but was not considered a time critical event. Additionally, programming and erasing the EEPROM required much higher voltages than the circuits that used the EEPROM, so external tools with higher voltages were used to program or erase the chip. There was a limit to the number of times a chip could be erased and programmed, ranging from several hundred up to around a thousand. Production-embedded devices rarely see more than several dozen program changes, so this limitation was not seen as a major issue.

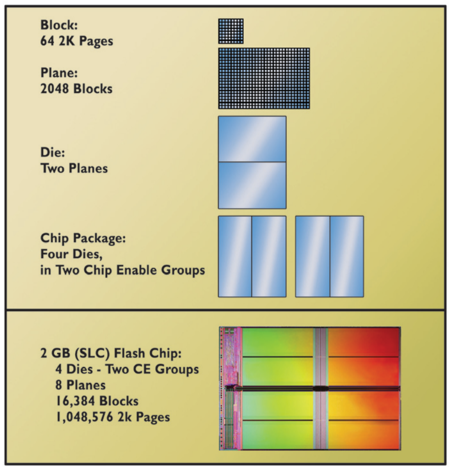

Figure 1. Anatomy of Flash

EEPROM is byte programmable – that is, any single byte can be erased and re-written independently of all the other bytes in the system. This required two transistors per bit, one to read the bit, and another to erase. In the late 1980s, engineers at Toshiba found a way to reduce costs by cutting the transistor count in such devices in half. This new circuit, called "flash" memory, uses one large transistor to erase all the bits in a "block" at one time. This allowed the number of transistors in a Flash memory circuit to be cut in half, giving proportional reductions in cost. One drawback of this technique is that data must be managed in blocks rather than as individual bytes. We will see later on how this can pose some challenges to the Flash user.

There are two kinds of Flash – NAND and NOR. We will not try to defend the reason these names have been applied to the two technologies, but will explain that NAND Flash reduces the number of connections that run across the chip to sense the state of each bit, and through this reduction shrink the chip size (and the cost) even further, allowing NAND chips to be manufactured at a fraction the cost of their NOR counterparts. A penalty for this shrink is that the data that is stored in the device is sometimes corrupted, and error correction algorithms must be used to scrub the errors out of the data. All NAND is used with a controller chip that handles this error correction, along with other housekeeping functions.

Chip Level Endurance

As noted in Figure 1, a 2GB SLC (Single Layer Cell) NAND Flash chip is composed of 4 dies, 8 planes, 16,384 blocks, and 1,048,576 2K pages. Each plane has its own block program/erase circuit and pool of blocks. To increase the capacity of a Flash chip without dramatically increasing the cost, four silicon Flash dies are packaged together and share a set of pins emanating from the chip. The four dies are divided into two chip enable (CE) groups; each group operates like a separate chip with two dies. This organization affects how read and write operations occur in a Flash chip.

Writing data to a Flash chip involves three main operations: read, erase, and program. Reads can occur randomly anywhere within the 16,384 blocks of a typical Flash chip. Erasing sets all of the bits in an individual Flash block to 1. Programming only occurs on blocks that are erased, because the program operation can only change a bit from a 1 to a 0. The high voltages needed for the erase operation originally gave Flash its name.

Flash drives targeted toward the consumer market in thumb drives, cameras, media players, etc., accept the limited endurance or write cycles of Flash and its poor write performance because writes occur fairly infrequently and therefore don't pose a problem. In the enterprise, however, the erase stress and endurance, coupled with slow write performance and some data integrity issues noted below, explain why until recently Flash drives were seldom deployed in mission critical enterprise environments.

SLC vs. MLC Flash

Up to this point, the discussion has concerned only SLC Flash. There is a second type of Flash called Multi-Layer Cell (MLC). The two types of Flash are very similar; in fact, their manufacturing processes are nearly identical. SLC is faster and more reliable, while MLC can store data more densely. Each holds a voltage to indicate a value stored in physically identical Flash cells. But SLC stores only two values, 1 or 0 (with either a high or low voltage level) while MLC can store four values (high, medium high, medium low, low) representing two bits per cell (00, 01, 10, 11).

Challenges of Using NAND Flash

As a storage medium, NAND Flash offers some unique challenges, including bit errors, wear-out, and part failures, particularly during infant mortality. The voltage required to set the value of a NAND Flash cell, combined with the ever-decreasing size of the cells, creates some probability that there will be errors, and that some cells will become unreliable with time and use.

Bit Errors

Errors can occur for a large number of reasons, many of them random occurrences, but the small (and ever decreasing) size of NAND Flash gates makes them particularly susceptible to error and failure. One such example is read or program disturbs. This is when reading or programming the data in one cell inadvertently disturbs the data in a nearby cell. NAND Flash errors can also be caused by elevated heat, manufacturing defects, or even simply repeated use, also known as wear out.

Wear Out

Another peculiarity of NAND Flash as a storage medium is its increasing susceptibility to bit errors after having gone through a certain number of program-erase (P/E) cycles. The number of cycles varies, depending on density (storage capacity per physical area), vendor, and NAND Flash type. Generally speaking, SLC is usually rated for ~100,000 P/E cycles, whereas MLC Flash is usually rated for ~5,000-10,000 P/E cycles. NAND Flash also becomes less reliable over time when unpowered.

Failure

NAND Flash storage is typically made up of a large number of chips, all of which are susceptible to higher rates of failure during their infant mortality period. Additionally, as with any electronic device, there is a possibility (albeit very small) of failure of any component part.

Read/Program/Erase Compensation

While NAND Flash allows very fast read times, programming and erasing data take comparatively longer. Programs take ten times longer than reads (hundreds of microseconds) and erases take roughly 100 times longer than reads (multiple milliseconds). Therefore, the current state of Solid State Storage (SSS) technology must deal with asymmetrical access. There are a wide variety of controller strategies to deal with this and improve performance.

Good SSS Controllers Bring Out the Best from NAND Flash

While NAND Flash may present some unique challenges, both the hardware and the software that manages SSS (which may be divided between the on-board controller, the driver, and other management utilities), when designed properly, can compensate for those challenges. Following are some of the ways that well-engineered SSS produces a high level of data integrity and availability.

Benefits of Robust Error Correction

Many of the inherent weaknesses of NAND Flash can be compensated for using robust error correction routines (called error correction codes, or ECC).

Uncorrected (Raw) Bit Error Rates for SLC and MLC

Figure 2 shows the bit error rate (BER) for both MLC and SLC NAND Flash for two different vendors. The BER is plotted as a function of the number of P/E cycles the block has undergone. The NAND Flash BER increases with the number of P/E cycles. As expected, the rate of increase of errors for MLC is substantially greater than for SLC. It is important to note that there is also substantial difference in both the rate of increase and the starting rate of errors between NAND Flash vendors.

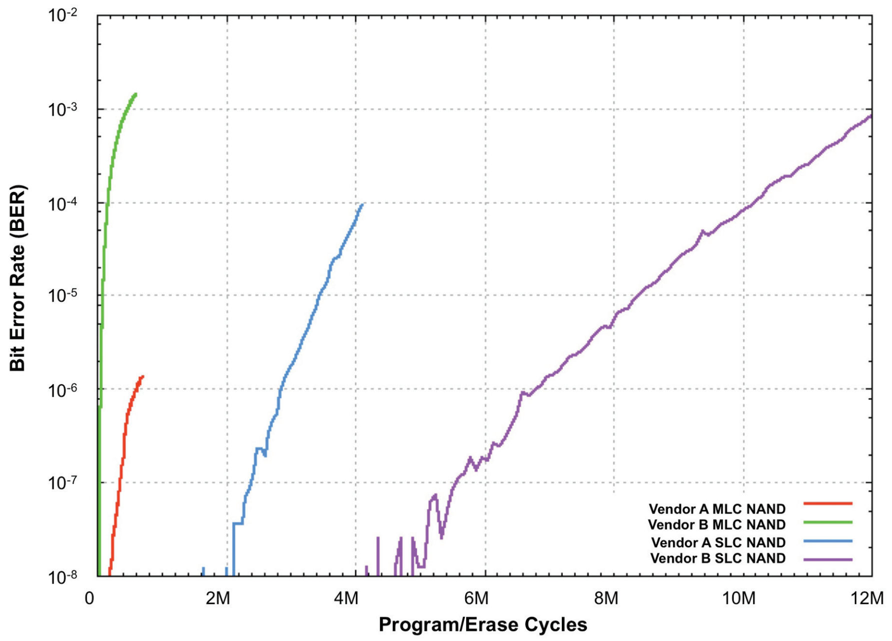

Figure 2. Bit Error Rates (BER) as a function of P/E cycles

There are several ways to reduce the BER level:

- ***Manufacturing Screening –*** it has been demonstrated that manufacturing screening can reduce the error rate by 2 to 4 times.
- ***Wear Leveling –*** to extend the useful life of SSS, it is essential to ensure that no specific area of the media receives more than the average number of P/E cycles.
- ***Write Amplification Avoidance –*** internal processes within the controller to manage the writing of data and wear leveling actually increase the number of writes to the media beyond the number being transferred from the client. This increase in the number of writes is called write amplification. Advanced methods that decrease write amplification can dramatically reduce the wear on the device and increase its usable life.
- ***Robust ECC –*** perhaps most important, advanced error correction codes as deployed in a variety of common media must be used to reduce the affects of this raw bit error rate.

Corrected Error Rates

While extension of usable life is one of the important benefits of applying strong error correction, it is perhaps more important to improve data integrity. Figure 3 illustrates the reduction on BER with various ECC schemes over a range of P/E cycles. The top line is the BER of NAND Flash without any error correction. The BER for unused NAND starts above 10-8 before any P/E cycles have been performed, and rises two orders of magnitude to 10-6 as it approaches 500,000 P/E cycles. With the application of 4-bit ECC (up to 4 errors can be corrected for every 512 bytes) the BER starts at a level over eight orders of magnitude lower than that of uncorrected NAND. Over several P/E cycles, however, the corrected data's error rate rises to a level close to that of uncorrected NAND before any P/E cycles have been performed. With a far stronger ECC algorithm, in which up to 11 errors out of 240 bytes are corrected, the BER for new NAND starts more than fifty orders of magnitude lower than that of uncorrected NAND and over 36 orders of magnitude lower than NAND with 4-bit error correction. As the NAND degrades with higher numbers of P/E cycles the 11-bit algorithm's BER rises to 10-20, or roughly that of the combination of 4-bit correction with unused NAND.

The horizontal and vertical arrows in Figure 3 show how aggressive ECC can either extend usable life or improve data integrity. Since there is some tradeoff between these goals, the capability of a particular solution's ECC capability and how this capability is applied—whether to reduce the effective bit error rate and therefore the data integrity, or to improve the life of the product—is vendor specific.

Robust ECC Substantially Extends SSS Useable Life

As mentioned above, P/E cycles are one of the principal causes of NAND Flash wear out. Typically, NAND Flash chips detect that a physical erase block (PEB) is unusable by detecting a bit that cannot be correctly erased. This is then indicated to the controller as an erase failure. The controller can then mark the PEB as retired and take it out of service, or it can choose to perform error correction as a work-around. For many devices, these bit erase failures do not represent the overall quality of the cells throughout the erase block, especially during the infant mortality period, so the possibility of performing error correction is a viable and useful option.

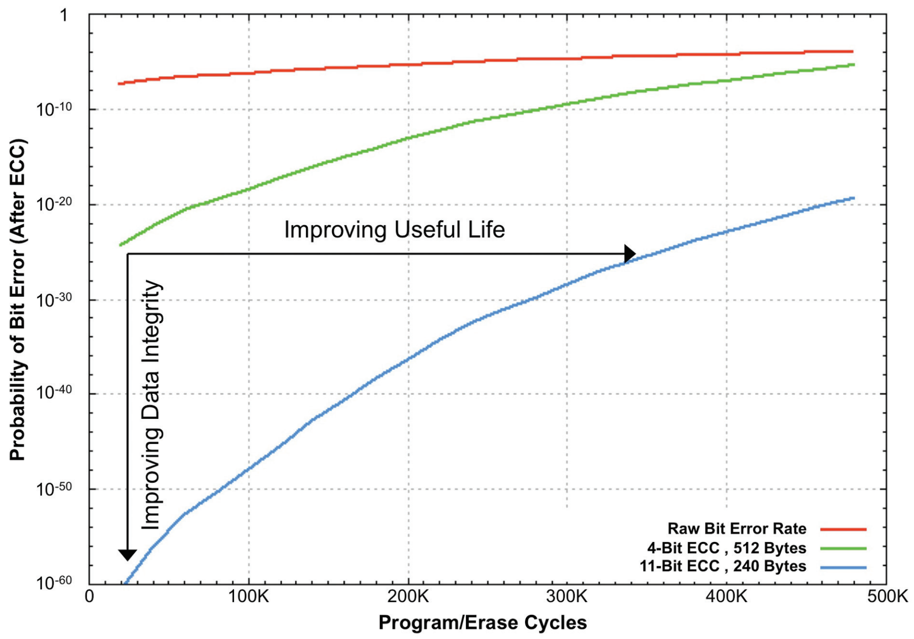

Figure 3. ECC improvement of raw BER as a function of P/E cycles

Figure 4 and Figure 5, illustrate the probability for both MLC and SLC Flash, that a PEB will contain one or more bit errors after a number of P/E cycles. This appears as the top green line and the bottom blue line in both figures. The red line in the middle is the same raw bit error rate as illustrated in Figure 3 and is shown only for reference. The PEB's BER is substantially greater than the probability of a single bit error (raw BER). This should not be surprising, since the probability of a bit error in an N-bit PEB should be N times the probability of a single bit failure. The fact that these lines diverge illustrates the effects of mechanisms other than program/erase cycles that cause the PEB's BER to be even more sensitive to the number of P/E cycles than the raw BER would indicate.

Some controllers simply decommission a PEB whenever a single bit error is recorded. The green line clearly shows that the probability of a PEB being removed from service is very high for this approach, and becomes certain (a probability of 1) after only 250,000 P/E cycles for MLC Flash (Figure 4), and about 3 million cycles for SLC Flash (Figure 5).

The bottom blue line illustrates the substantial improvement gained through the use of robust ECC. When robust ECC is introduced, it is possible to ignore single errors within a PEB. Some controller architectures even distribute data across multiple chips and therefore multiple physical erase blocks to further reduce the effects of bit errors in any one block. Depending on the controller architecture, it may be possible to keep an erase block in service with known bad bits. As an example, in Figure 4, an "aggressive" ECC algorithm that has the ability to correct 11 bit errors in an ECC code word improves the useable life of an MLC PEB by roughly an order of magnitude. Figure 5 shows this same code significantly improving the useable life of an SLC PEB.

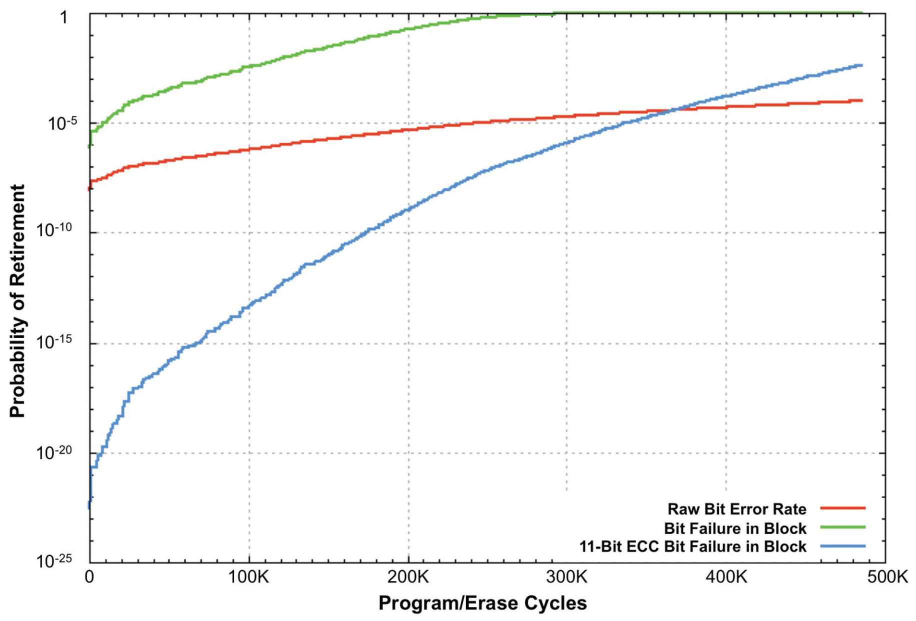

Figure 4. Probability of PEB retirement in an MLC NAND Flash SSS device

Read Disturbs and Program–Erase Cycles

Read disturbs were mentioned earlier as one of the challenges of using NAND Flash. When a bit is read, there is a chance that it can change the state of an adjacent bit. The more times a bit is read, the greater the likelihood that it will flip an adjacent bit. As with other bit errors, the probability that a read disturb will occur also depends upon the number of P/E cycles that a block has experienced.

Typical read disturb probabilities can be seen in Figure 6. The chart gives the probability of a bit error as a function of the number of read cycles for four different levels of P/E cycles. These curves depict statistics for an MLC Flash chip, which is more sensitive to this phenomenon than is SLC Flash. The bottom red line shows that read disturbs can occur before 8 million read cycles in a chip that has undergone a very low number of P/E cycles. The next higher line (in green) shows that as the number of P/E cycles increases to only 2,000 the probability of a read disturb increases significantly, and even low numbers of reads can cause a probability of about 2*10-8 of a read disturb in any one bit. Please note that this level of 2,000 P/E cycles is a much lower number than shown in any of the previous figures.

As the number of P/E cycles reaches 6,000, the top curve shows that the probably of a read disturb error increases to a full three orders of magnitude higher than that of the zero P/E condition. Read disturbs are clearly very sensitive to the number of P/E cycles the block has handled.

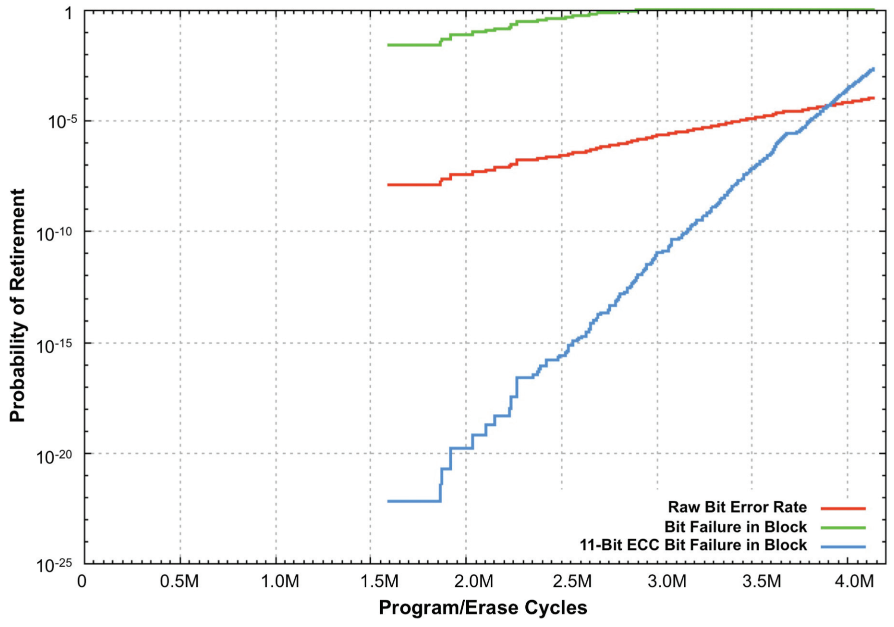

Figure 5. Probability of PEB retirement in an MLC NAND Flash SSS device

Generally, Flash manufacturers specify read disturb data that has been measured on cells that have not undergone previous P/E cycles (corresponding to the bottom red line on Figure 6: "Read BER after 0 P/E cycles"). We have just seen, though, that the number of read cycles that can be achieved without introducing data errors is dramatically reduced after a few thousand P/E cycles and further decreases with use. This underscores the need for robust ECC. Another preventive measure is to "scrub" the data by periodically reading it from the NAND, correcting it with ECC, and writing it back into the device, somewhat similar to the process of refreshing a DRAM. Although scrubbing increases the number of P/E cycles, it is required to maintain the integrity of the data. Whereas this and other environmental and operational effects can cause wide variations in the rate of read disturb errors, it is important for controller vendors to scrub only as often as necessary. Additionally, some users may benefit from being able to dynamically adjust scrubbing parameters to match their use cases.

Thermal Protections

One of the strengths of NAND Flash as a storage medium is the lack of moving parts that can generate heat. Nevertheless, all electronic devices generate some heat, and must be able to cope with the ambient heat of other parts of a computer, so it's very important for an SSS device to provide adequate thermal protections, allowing for or potentially generating adequate airflow to mitigate the heat it generates.

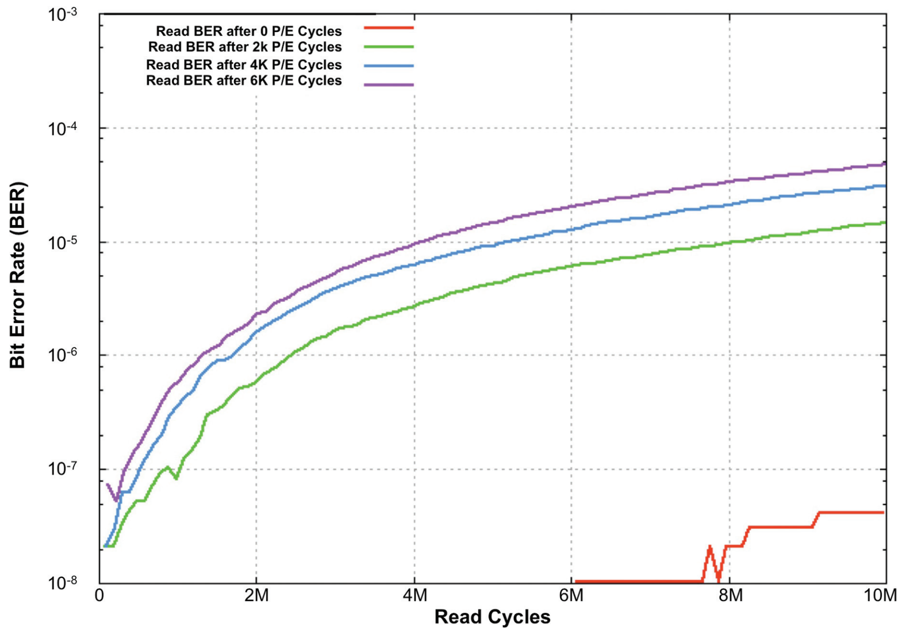

Figure 6. Probability of uncorrected bit errors in a sample MLC media as a function of number of read cycles

Power Loss Protection

Good storage devices must never tell the system they have committed data until it has actually been committed, to avoid losing the data after an unanticipated loss of power. From a performance standpoint, since NAND Flash programs take a comparatively longer time (in computer time terms), this is a potential management challenge.

Retirement Strategy

Since NAND Flash is susceptible to infant mortality failure in the short term and wear out in the long term, SSS devices must be able to recognize less reliable parts of the storage medium and retire them as appropriate. Ideally, the device should be able to retire storage chunks with variable granularity, to avoid unnecessarily retiring usable storage space.

Wear Leveling

In order for SSS to be relied on for a long and useful life, it must be able to compensate for the fact that NAND Flash can only withstand a certain number of P/E cycles. In other words, it can't simply continue to use one small portion of the storage medium and wear that portion out, since this strategy would constantly reduce either the reliability (if sections receiving too much wear are unretired) or capacity (if those sections are retired) of the SSS device. Therefore, good SSS management involves ensuring that the data is spread out evenly across the storage space over time. This is typically accomplished both at the time the data is initially programmed (referred to as dynamic wear leveling) and later, as part of the management of the data, to provide wear leveling across the entire storage space (referred to as static wear leveling).

Data and Management Metadata Protection

A previous section of this paper described how robust ECC can greatly improve data integrity even when there are bit errors, but strong ECC is not the only way to ensure data integrity. Some vendors also use an approach that implements a data integrity field (DIF). DIFs, which are similar to checksums, are appended to data segments stored on the SSS to protect against silent data corruption. Some SSS devices also use a form of internal RAID to both check for and repair corrupt data.

Finally, it's not enough to protect only the data. Application data is typically accompanied by some kind of management metadata (akin to the logical block address to physical block address maps for traditional drives). The management metadata must be protected at least as well as the data itself to ensure data integrity.

Avoiding Partial Page Programs and Write Amplification

Some SSS data management strategies designed to solve one kind of problem can lead to other kinds of problems. For example, some controllers program data as it is received, even if its smaller than a full erase block, a method called partial page programming. When new data must be written, the old data must be read and stored in a buffer while the block is erased, then combined with new data, then rewritten. Depending on the size of the chunks being written, partial page programming can cause significantly more writes, or write amplification, because of the larger number of P/E cycles required to write new data. An alternative strategy is to hold the data in a buffer until a full page has been received, and only then program the data.

Another cause of write amplification is mis-tuning; i.e., not being able to tune an SSS device to write data in block sizes that are a good match for the applications using the storage. The average size of files being written and read/write ratios strongly influence how an SSS device should be configured, to avoid having to produce unnecessary P/E cycles to consolidate data and perform garbage collection.

Transport Errors

Besides protecting data that has been programmed (i.e., at rest), data must be protected as it makes its way between the host system and the SSS storage medium (i.e. in flight). Data that is in flight must also be checked and protected against the possibility of silent data corruption. Ordinarily, this is handled via mechanisms specified in industry standards, such as SECDED (single error correction, double error detection), chipkill (advanced ECC that protects against single memory chip failure as well as multi-bit errors from any portion of a single memory chip), 8B/10B (a powerful encode/decode scheme), parity, and checksums.

Mitigating Pathological Write

Another potential problem that can result from improper configuration is a condition called "pathological write." With NAND Flash technology, there is no single "write" operation: writes are executed by performing an erase operation followed by a program operation. As was mentioned earlier, Flash is slow to program and even slower to erase.

SSS Vendors compensate for the slower program-erase operations using creative "grooming" methods. These include techniques like erasing bigger set of bits at a time, and doing so well ahead of the need to program them, making the program-erase operation transparent to performance. These are usually performed by the controller as background operations to assure that a write operation will be met with an available, erased space. The background operations are performed on spare blocks of Flash that are not made visible to the computing environment. But, given enough "write" activity, the groomer will eventually fail to provide a sufficient pool of pre-erased blocks, slowing down the program operations. This condition is referred to as a "pathological write."

Figure 7 shows the effect of pathological writes. In this figure the maximum bandwidth that an SSS can absorb is plotted as a function of time - the duration of a series of pathological writes. To get these results, writes were performed non-stop to random locations that spanned the address space of the drive. Each of the lines represents a different set-up, with the first five lines representing an 80GB PCI device that can be configured by the user to present varying amounts of storage to the computing environment, reserving the balance for background operations (i.e. the top line indicates that only 30GB of the 80GB available was presented to the computing environment, with the other 50GB devoted to background tasks.) The bottom line shows a 30GB SATA drive that is set to offer 28GB to the computing environment, keeping 2GB aside for background operations.

All configurations clearly decline in performance after receiving hundreds of seconds' worth of pathological writes. The chart makes it clear that the severity of this performance degradation is a function of the amount of spare space that is reserved for background operations—the larger the spare space, the faster the pathological writes can be handled. As the chart shows, a drive's performance increases in accordance with the amount of space that is reserved for background operations.

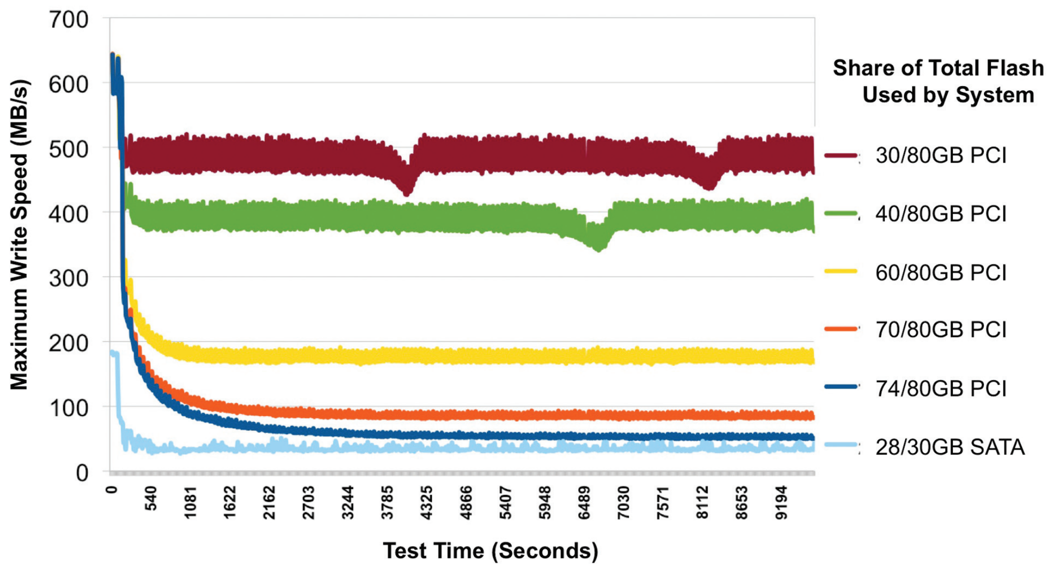

Figure 7. The impact of a pathological write condition, with and without reserve space tuning.

This "pathological" state is atypical of most real-world use environments as it requires write operations that are 1) continuous, 2) completely randomized, and 3) spanning a majority of the drive. Normally, intense writing occurs on a small fraction of the files or drive, and while the blocks comprising those files may be random, they are allocated and de-allocated together (which, for SSS, is not random at all). However, there do exist a small number of real-world applications that exhibit this behavior. In those rare cases, reserve capacity tuning can greatly mitigate the impact of the pathological write condition.

Graceful End of Life

Finally, one of the greatest advantages of NAND Flash-based SSS, from a reliability standpoint, as compared to rotating mechanical drives, is its ability to gracefully predict and manage its end of life. Whereas mechanical drives typically fail only catastrophically, NAND Flash SSS devices can anticipate failures, move and correct data as necessary, retire unreliable parts of storage media, and generally reach the end of their useful lives gracefully, having given plenty of advance warning. The ability to anticipate wear out is a natural consequence of the fact that blocks are expected to be decommissioned from time to time. Since the controller always knows how many blocks have been decommissioned and how many spare blocks remain, it can notify the user when the SSS is expected to run out of spare blocks well in advance of this occurrence. Replacing SSS storage devices is far more likely to be a matter of routine maintenance than replacing rotating, magnetic drives, which, as numerous studies have shown, frequently fail with little or no warning.

Reliability Feature List

NAND Flash SSS can provide world-class performance and reliability at the same time, but not all solid state storage is designed to meet all needs. Therefore, before you buy, you should consider how great your need is for reliability (both integrity and availability) and shop accordingly. Following is a summary of some reliability issues to consider and ask of vendors:

- Data protection
    - Robust ECC
    - Internal RAID
    - In-flight data protection (SECDED, chipkill, 8B/10B, etc.)
    - DIF
- Wear leveling
- Thermal protection
- Power loss protection
- Partial page program avoidance
- Intelligent and granular memory retirement
- Block size tuning
- Reserve space tuning

Conclusion

NAND Flash is already used widely and successfully in consumer electronics. Its adoption in enterprise environments, however, has been slower (though growing rapidly), in part due to a perception that NAND Flash as a storage medium is less reliable and durable. As this paper has shown, however, a good controller can more than compensate for the peculiarities of the medium itself while bringing out its primary strengths: speed, lack of mechanical parts, and comparatively low energy use. Properly managed, NAND Flash-based SSS isn't simply a performance solution: it's a performance solution you can trust with your most mission critical data.

SNIA

Storage Networking Industry Association

425 Market Street, Suite 1020 • San Francisco, CA 94105 • Phone: 415-402-0006 • Mobile: 415-420-2145 • Fax: 415-402-0009 • www.snia.org