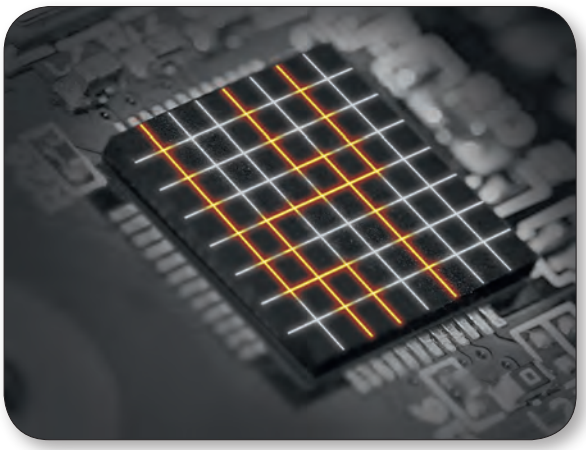
New flash management architecture enables MLC for industrial storage

By Susan Heidrich, Hyperstone

This article describes hyMap™ flash management architecture, a new sub-page-based mapping and flash translation layer approach which significantly improves write amplification, increases endurance and random write performance. A low random write WAF results in high random write IOPS, preventing stress on the flash and giving the flash device a longer life.



■ Flash memory is one of the most important storage technologies in terms of keeping and quickly accessing large data. This refers to the consumer market as well as to industrial applications. For storing and accessing data on a flash, a storage device controller is necessary. However, not only the controller hardware, but especially the dedicated firmware with complex algorithms enable safe flash handling and reliable data management and allow for market differentiation.

File systems use certain access patterns for writing to storage systems. The smallest access unit usually is a sector of 512 bytes. 4 Kbytes is a rather common unit, but it can be up to 256 Kbytes. There are several different protocol types such as ATA, USB, SD or MMC that pass read and write commands either directly or queued to a storage device controller. In all flash storage systems, a flash translations layer (FTL) including logical to physical mapping is applied that translates host-side logical to flash-side physical accesses. According to individual application requirements in terms of cost, performance and other desired features, different mapping approaches and granularities can be applied. The new hyMap™ technology is targeted to guarantee maximum reliability and endurance while improving random performance significantly.

Finer mapping granularity

The smallest read access unit of a NAND flash can be a sector (512 bytes). The smallest write access unit is one page. Page sizes differ among flashes and can be between 2K to 8K for SLC and 8K to 16K for MLC today. Another flash-specific feature is the fact that a block must be erased before pages can be rewritten. Also block sizes differ depending on flash technology and capacity and can range from 128 KB to 8 MB.

Still mostly used within consumer USB flash drives or SD cards, is "block-based mapping". In this approach logical blocks are mapped to physical blocks. Pages within those blocks are directly 1:1 allocated between the host view and the flash view. This means logical page 1 within any block refers to physical page 1 in the associated physical block.

Another approach is the "page-based mapping" in which logical pages are mapped to physical pages. Block numbers are part of the page address therefore a logical page can be mapped to any page within any block.

An even finer mapping approach is "sub-page-based mapping" where logical units smaller than a page are mapped to physical units. However, as the smallest possible unit which can be written to a flash is still one page, those units need to be consolidated to one page. When reading that smaller unit the controller does not need to transfer the whole page from the flash into the controller SRAM. The finer the granularity of the mapping, the better it can be tuned to different usage models. hyMap™ is sub-page-based by default and can be compiled to different granularities.

Very low RAM requirements and inherent power fail safety

The finer the granularity of the mapping, the more complex is its algorithm. Apart from providing more computational power, more mapping information must be stored when updating, and retrieved when reading. Historically, most SSDs have stored all mapping information in an external DRAM. While this is convenient and fast, it is very difficult to ensure power-fail robustness. The average size of mapping data for a conventional page-based mapping FTL is in a range of 0.1% of the drive capacity e.g. 32 MB for 32 GB. For a sub-page-based mapping it is even higher. Every time a drive shuts down, this DRAM content needs to be stored in non-volatile memory (NVM). For stable operations, super caps need to be added to ensure power supply in case of a power loss. Apart from cost, this introduces another element that would need to be considered when assessing drive

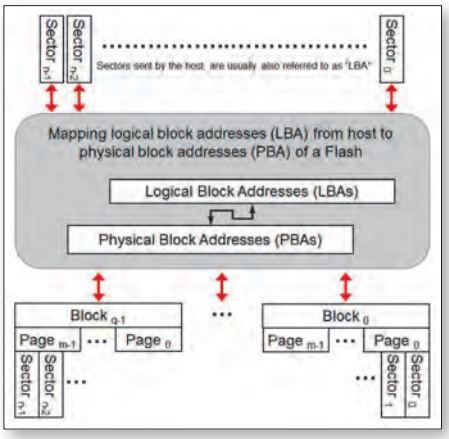


Figure 1. Basic block-based mapping

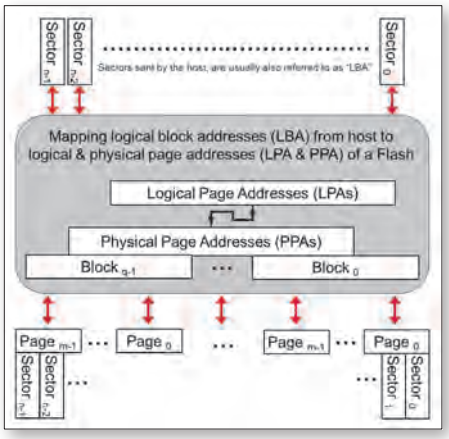


Figure 2. Basic page-based mapping

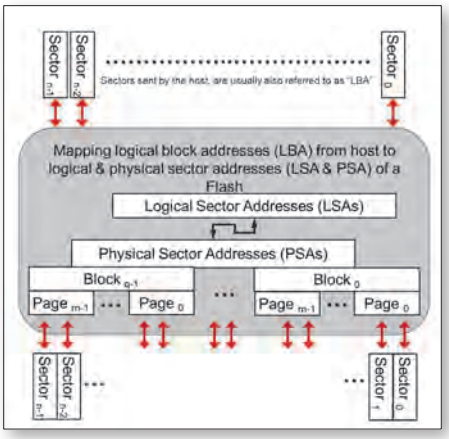


Figure 3. Basic sub-page-based mapping

reliability, life-time estimation, MTBF, and power on-off cycles. hyMap™ continuously updates necessary mapping information in the NVM by using a sophisticated algorithm and transaction-oriented logbook. Therefore, power-fail safety is ensured at all times.

Lower random-write WAF

The write amplification factor (WAF) refers to the amount of additional information that must be written to a flash when a certain amount of user data is written to the flash.

$$\text{WAF} = \frac{\text{Bytes written to the Flash}}{\text{Bytes written by the host}}$$

For instance, a WAF of 4 related to 4K random writes would mean, that for each 4 Kbytes of data in fact 16 KB are written to the flash, if blocks must be erased to free-up capacity and consolidate unused pages distributed over different blocks. This means that one 4 Kbytes write causes a block-erase and several page-writes to this block. Calculating the WAF is complex and depends on many different factors including: mapping granularity, efficiency of mapping and garbage collection, flash technology (SLC/MLC, block size, page size), caching of data, storage location of mapping data (external DRAM or NVM), command queuing if protocol permits, data access/write patterns, free capacity on the drive and organization of that free user space (TRIM), over-provisioning, and data integrity requirements of the host or file system and possibility to use command queuing and write data caching to reduce WAF. For block-based mapping, write amplification can be significant, especially for small and random write accesses. A single-sector-write under worst case conditions might require that a whole block is erased and all original content is copied and stored together with the single updated sector within that block.

In such a scenario, the WAF would be as high as 2000, since for a 512 byte sector it would be necessary to write a whole block of 1 MB. For sub-page-based mapping such worst case write amplification is reduced significantly. A single-sector-write under worst case conditions might also require that a whole block is erased in case no unused block would be available but no old data would need to be copied. Only the old sector would become invalid and the new sector in another block would become valid. In order to program a single sector under worst case conditions, it might be required to program the whole page. In that case the write amplification would be 16 (8 Kbyte / 512 byte).

Comparing both mapping approaches with respect to worst case WAF for write accesses smaller than page size it becomes clear that a finer mapping has significant merits and would boost drive endurance by a factor close to 100 times. In most use cases, accesses are not only single sector writes and also the controller can very often write more sectors at each time to the flash. Enterprise workloads are heavy on 4K random writes. This might not reflect a regular system usage model. So, for mainly sequential accesses or streaming applications, payload sizes would probably shift more towards larger sizes while for embedded OS, status updates, or communication protocol related tasks payload sizes might shift to smaller payload sizes. For usage models with a mix of payload sizes it is quite difficult to calculate exact WAF figures but as an order of magnitude WAF of the sub-page based mapping might be over 30 times better.

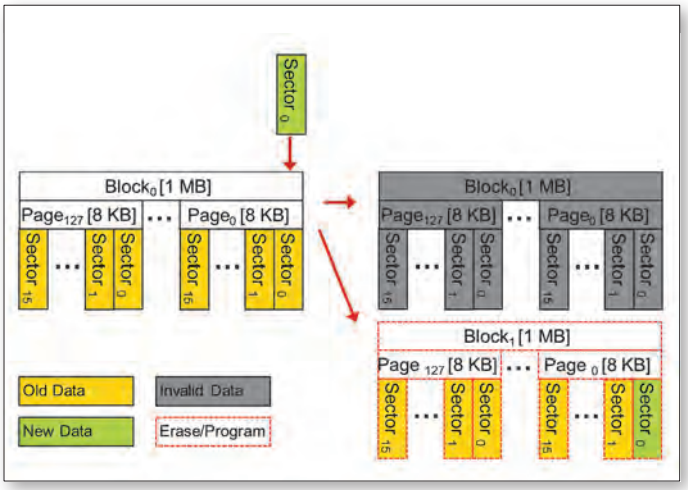


Figure 4. Worst case write amplification for an example access condition (random single sector write) for block-based mapping

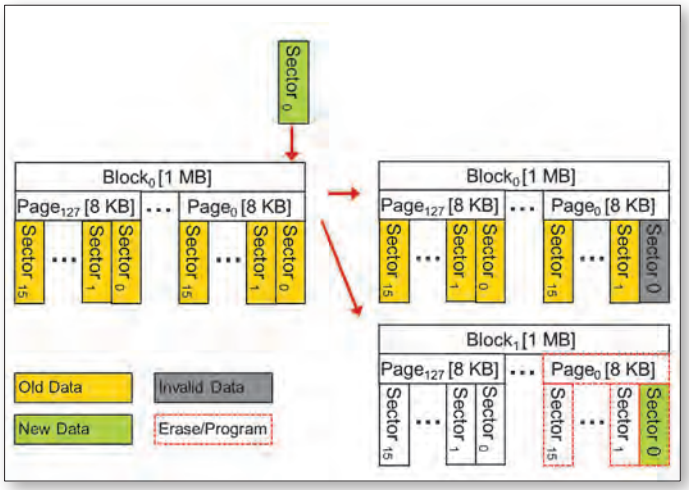


Figure 5. Worst case write amplification for an example access condition (random single sector write) for page-based mapping

		Block Based Mapping		Hyperstone hyMap™	
		Read (MB/s)	Write (MB/s)	Read (MB/s)	Write (MB/s)
SLC	Sequentiell	95	60	90	60
	4K Random	5.7	0.09 (20 IOPS)	3.1	4.0 (1000 IOPS)
MLC	Sequentiell	75	25	70	25
	4K Random	4.7	0.03 (7 IOPS)	3.3	3.9 (975 IOPS)
SLC: 2x nm, 8K page, 100 MHz DDR I/F, 2 CE DDP MLC: 1x nm, 16K page, 100 MHz DDR I/F, 2 CE DDP					

Table 1. Performance comparison for an SD card in SD3.0 UHS-I mode using S8 with block-based mapping and with hyMap™ firmware

Enabling MLC for industrial / embedded applications

Knowing the WAF associated with an individual usage model and drive pre-conditioning enables the calculation of the overall drive endurance expressed in Terabytes Written (TBW) and rate drives.

$$TBW = \frac{Capacity[GB]}{1000} x \frac{P/E\ Cycles}{WAF}$$

Considering a scenario in which an individual usage model WAF would be in the range of 5 for hyMap™ and 200 for a block-based-mapping architecture, it can be affordable to replace an SLC flash with 100K write-erase cycles by an MLC, providing 3K write erase cycles and getting the same endurance in terms of TBW. That means, the better the mapping granularity is tuned to an individual system access pattern, the lower the WAF and the higher its drive endurance. hyMap™ can save money if it is found that now MLC can be used where SLC was required before.

Performance optimization

At some point physical blocks containing sectors or pages with obsolete data must be consolidated to free-up storage space. This process is called garbage collection (GC). An efficient GC algorithm selects those blocks first that contain most obsolete pages. The process is either performed in the background or during idle times. The availability of excess storage capacity compared to the net capacity of the logical data space is referred to as over-provisioning (OP). This is necessary to provide extra blocks so that GC can consolidate fragmented valid data stored in several used blocks. Also, depending on the amount of OP space, GC can work more efficiently as the share of valid data compared to invalid data of used blocks decreases, less data has to be copied to new locations and more free space can be provided per operation. hyMap™ can utilize available NVM resources to improve performance and reduce WAF by dynamically using unused capacity for over-provisioning.

The effects of hyMap™ in terms of performance are shown in table 1.

Data reliability and device-life extending features

hyMap™ uses a patented wear levelling (WL) algorithm that can be configured to the needs of different flash technologies as well as application requirements. WL is used to systematically utilize all flash blocks of the system equally in terms of consuming their individual write-erase-cycle endurance budget. It supports dynamic, static, and global wear levelling.

Dynamic WL requires no copy-overhead but alone would be limited to blocks not containing data. Static WL includes also those blocks containing data. Static data is relocated if needed. This WL activity is triggered at predefined threshold levels. Also these routines are executed in the background and interrupted in case of higher priority host commands.

Power Fail Robustness without external DRAM

As soon as a power-down is recognized, the controller is reset and the flash is write-protected. A log of all recent flash transactions is kept. Should the latest data be corrupt, the controller will recover the latest valid entry before that last failed write. All mapping information is reliably stored on the flash and therefore external DRAM is not needed.

The new firmware concept is targeted to making MLC flash as reliable as possible. As two logical MLC flash pages are physically correlated, it is possible to destroy data of an older page by writing another new one within the same block (paired pages). hyMap™ applies a Reliable Write feature to cope with this occurrence to make MLC power-fail safe. Moreover, it uses Safe Flash Handling in unstable power supply situations and especially when exposed to a sudden power-down in which the last programming of pages might not be reliable although a flash might have reported successful writing. ■