efficiency embedded

White Paper Flash Controller & Firmware

Version 1d | May 05, 2008

Hyperstone GmbH
Line-Eid-Strasse 3
78467 Konstanz
Germany
Phone: +49 7531 98030
Fax: +49 7531 51725
Email: info@hyperstone.de

Hyperstone Inc. - USA
465 Corporate Square Drive
Winston-Salem, NC 27105
USA
Phone: +1 336 744 0724
Fax: +1 336 744 5054
Email: us.sales@hyperstone.com

Hyperstone Asia Pacific - Taiwan
3F., No. 501, Sec.2, Tiding Blvd.
Neihu District, Taipei City 114
Taiwan, R.O.C.
Phone: +886 2 8751 0203
Fax: +886 2 8797 2321
Email: taiwan@hyperstone.com

www.hyperstone.com

hyperstone®

Solid State Discovered – Controllers in a Flash

Hyperstone flash memory card or solid state disk controllers together with included firmware provide considerable mechanisms when handling a complex medium – making flash memory easy to use for consumer applications and reliable for industrial applications. Furthermore, firmware that is stored in the flash itself offers several possibilities for custom features, even when based on identical hardware.

Introduction

Clearly, flash memory has significantly replaced various historic storage media. As examples, some 5 years back people were using film in non-digital cameras, computers were offering floppy disk drives, portable CD players played your favorite tunes, and mobile phones were incapable of taking pictures, browsing the web or synchronizing with a PC. In recent years, consumers have become accustomed to using a large variety of NAND flash memory devices for data storage applications. Users of mobile phones, digital cameras, and other gadgets are familiar with the terms SD card, USB-stick, CF cards and the like. Such kinds of small and handy memory systems have become everyday commodities among consumers.

Requirements for such consumer applications are normally determined by

- Cost
- Compatibility to a plethora of host devices
- Fast transfer rates for reading and writing of data
- Storage capacity
- Conformity to standards
- Ease of use

Memory cards or USB sticks that fulfill these requirements can meanwhile be found in almost any supermarket or grocery store around the world. As the above features are not too difficult to obtain, it has become a challenge to list the variety of vendors who manufacture or sell such memory systems.

Have you ever experienced the sharp sound of your hard disk when it crashes? If so, you may specifically like the features of a solid state disk (SSD). Flash based solutions offer many advantages such as being faster, more power efficient, more rugged than rotating media, and being more easily integrated together with other chips in system design and production flows. Tremendous technology advances have decreased manufacturing costs and reduced prices significantly. These advances however are more and more demanding for very intelligent control functions.

Requirements for highly reliable flash memory storage systems, especially those not removable, such as SSD or embedded Flash or eMMC, are more demanding than for removable consumer grade cards:

- Good mechanism to detect and correct errors inherent to flash memories
- Efficient algorithms to maximize lifetime
- Tools to predict lifetime or monitor systems' status and health
- Ability to implement customer-specific features

Introduction to NAND Flash Memory

In many implementations, memory cells are based on CMOS floating-gate transistors. Each cell can represent one or more bits by reading out one or multiple levels of its electrical charge at the word line. This charge is changed using the Fowler-Nordheim tunneling or tunnel effect. Electrons are removed from or trapped in the floating gate. By applying a positive potential to either the bit line or the word line, the charge is changed and the cell is either erased or programmed respectively.

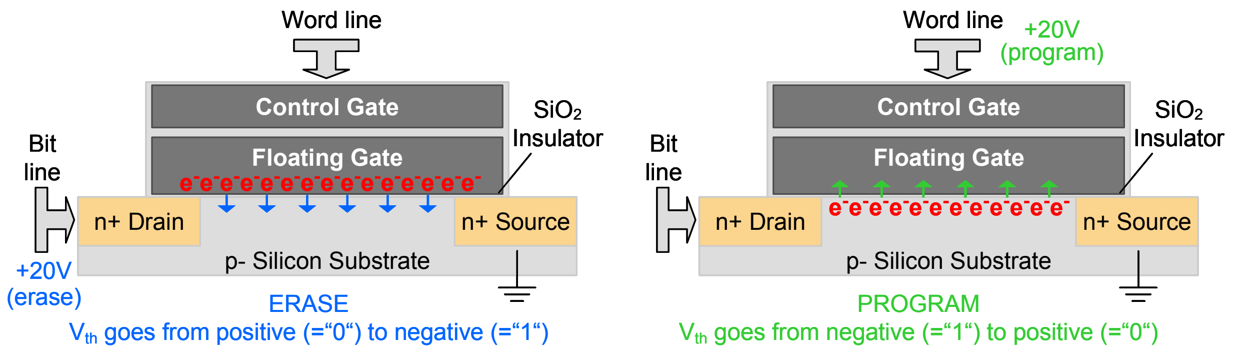


Figure 1: Program & Erase for an Example of a Floating Gate Memory Cell

So called Single-Level Cells (SLC) store or represent one bit. Charges (or threshold voltages V_{th}) between about -1V and -3V could be considered as representing the value "1" and charges between +1V and +3V as value "0".

So called Multi-Level Cells (MLC) store or represent two or more bits of information per cell. While the erase process and the erased level for Multi Level Cells is similar to that for SLC, for the programming or programmed state different charge levels of the floating gate have to be achieved. This is performed by applying Incremental Step Pulse Programming (ISPP). Again a positive potential in the range of 15 to 25 V is applied to a word line, but in short pulses. After each pulse a check is made to determine whether a desired charge level is reached. As soon as the desired charge has been reached and confirmed, the cell is considered programmed and represents two (or more) bits of logical information. Therefore, the writing process of MLC takes a little longer.

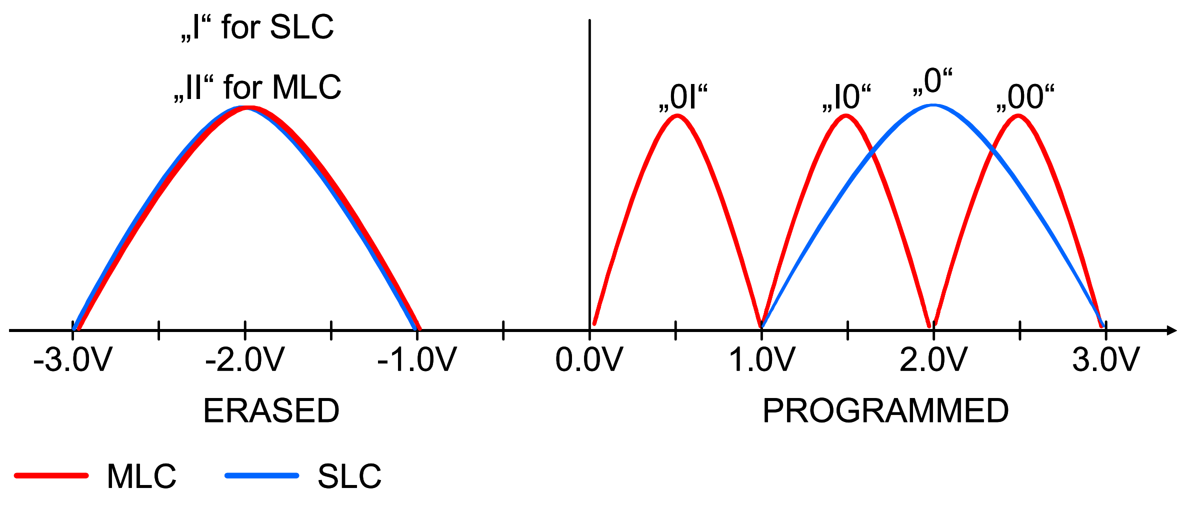
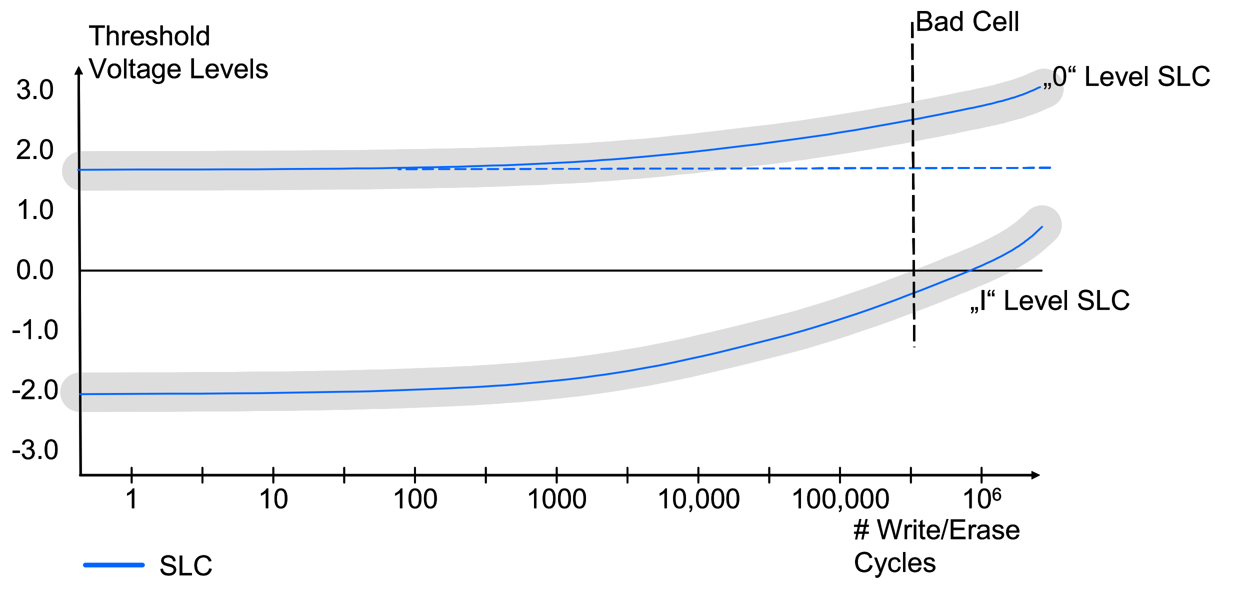


Figure 2: Example of possible voltage levels and related logical bit value of SLC and MLC compared

Writing and erasing imposes stress on the individual cell so that cells age or wear down over program/erase cycles. In effect, the charge levels change over time. While programming time decreases, erasing takes an increasing amount of time. Eventually charge levels exceed the defined thresholds and errors occur. As thresholds for MLC must be closer together, production variances have a greater effect and errors occur more readily and/or earlier over time.



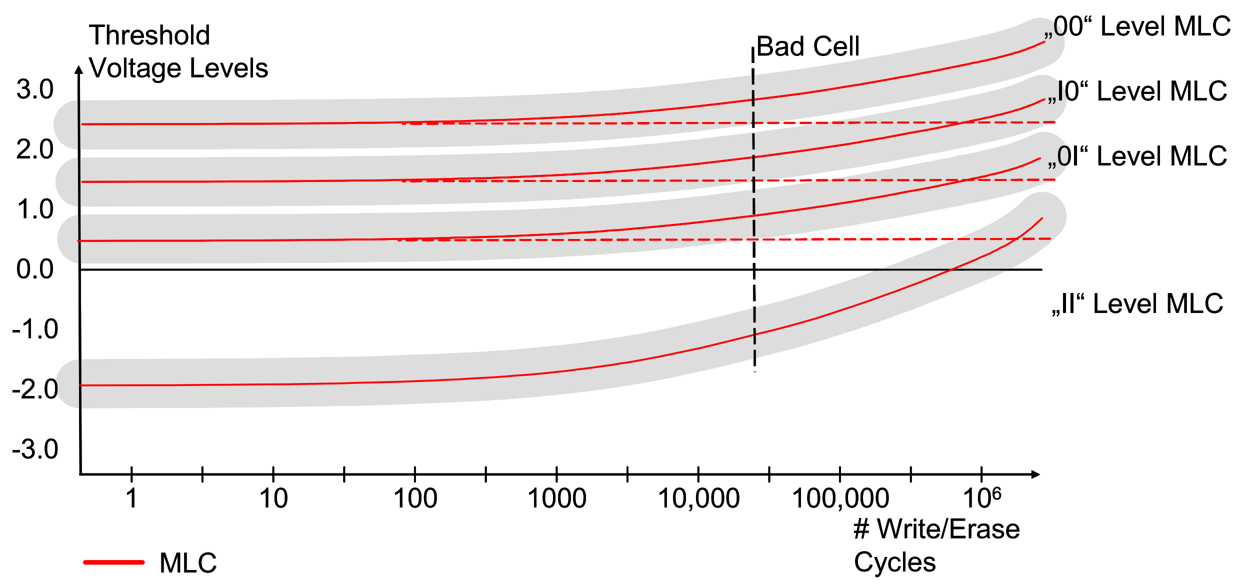


Figure 3: Example of possible voltage levels for SLC and MLC changing over time

Additionally, charge levels might change due to external conditions such as extreme heat or magnetism. While the cell itself is not damaged permanently, the bit value might have changed and a read error might occur.

Some more recent flashes have the capability of recognizing the systematic change in behavior or in the voltage level so that not the difference to a starting reference voltage, but the inability to differentiate the relative difference to other voltage levels produces the read errors.

Understanding technical limitations such as those resulting from process shrinks, it is also important to know that effects such as capacitive-coupling of internal signals and floating-gate-to-floating-gate noise can cause the cell's threshold voltage to shift by programming neighboring cells. This is one of the reasons that many believe there to be a physical barrier to shrinking geometries in the 30 nm region or slightly below.

Data Retention

While a cell's "quality" over time is mainly dependent on the number of write/erase cycles, the quality of a cell's voltage level or the data retention might also deteriorate over time. Often flash data sheets specify data retention in the area of 10 years, whilst real-life data retention might be subject to all kinds of intrinsic factors such as heat, magnetism, traffic on neighboring cells and more. On the positive side, remapping or wear leveling might improve the data retention, as data is actually moving across physical blocks and the last time a bit is physically written differs from the last time it was logically written.

Comparison to NOR Flash Memory

Cells in NAND flash are arranged in arrays of between 8 and 32 cells. Unlike in NOR flash, the individual cells are not connected to the bit line. For this reason, NOR flash requires more area and is slower to program and erase, but on the positive side, NOR flash achieves better random access times and can be programmed by byte.

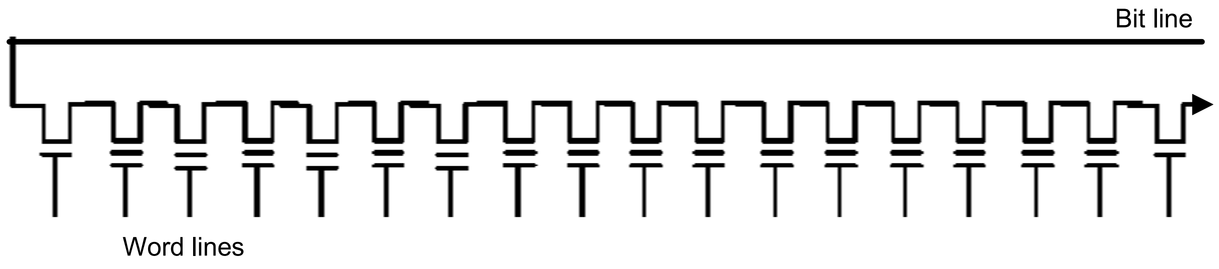


Figure 4: NAND flash array of 16 cells

Flash Memory Organization

When looking at how a flash memory is organized the smallest logical/administrative unit is a sector. Each sector contains 512 bytes plus an overhead area (traditionally 16 bytes). One to eight sectors are then grouped into pages, in the range of 512 to 4,096 bytes per page. At the next level of hierarchy blocks can include 32 pages of 512 bytes for example, or more recently 64 pages of 2,048 bytes. Blocks therefore also have a defined number of sectors, currently between 16 and 512 and there are 1 to 8 thousand blocks per chip.

Physical Block Addresses																										
Block 0									Block 1									Block n								
Page n			Page 1			Page 0			Page n			Page 1			Page 0			Page n			Page 1			Page 0		
Sector 0	Sector 1	Sector n	Sector 0	Sector 1	Sector n	Sector 0	Sector 1	Sector n	Sector 0	Sector 1	Sector n	Sector 0	Sector 1	Sector n	Sector 0	Sector 1	Sector n	Sector 0	Sector 1	Sector n	Sector 0	Sector 1	Sector n	Sector 0	Sector 1	Sector n

Figure 5: Organization of a Flash Memory

Writing or programming is done at the page level. Also, programming requires pages to be pre-erased. Once pre-erased, programming or writing can be performed one page at a time or by addressing a sector within a page that is pre-erased or "empty". Depending on flash memory type, pages can be accessed up to 4 times, when writing a sector.

Erasing is performed at the block level. As blocks are the "management" or "administrative" units, blocks will wear out as memory cells break down after a number of erase cycles. When defective, these blocks will be considered "bad blocks".

Flash memories from different manufacturers vary greatly with respect to bus width, number of blocks, block size, page size, die count, and programming capabilities, including caches. The following figure shows the organization of a typical NAND flash device available today.

Example Memory organization:		
Sector Size	512	Byte
Sectors/Page	8	
Pages/Block	64	
Page Size	4	KByte
Block Size	256	KByte
Blocks / Die	4096	
Dies/ Chip	2	
Flash Capacity	2	GByte

Figure 6: Example Organization of a Flash Memory (16 Gbit, dual die, SLC, 4K pages)

All of these characteristics pose some interesting challenges to the controller and its firmware:

- How to re-write to areas already containing data?
- How to maximize product lifetime with only a limited amount of erase cycles available?
- How to ensure data transfer integrity?

Flash Memory Controller and Firmware

The complex nature of flash cells and their organization demands reliable, high performance control functionality. Flash Controllers of all kinds consist of an interface to the flash memory, a processor and a host interface.

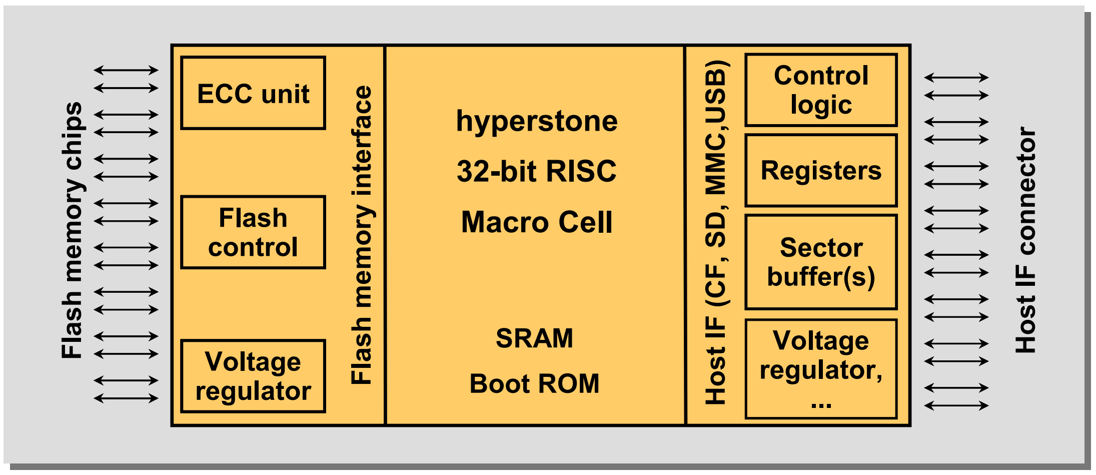


Figure 7: Generic Block Diagram Solid State Memory Solution

Hyperstone's controllers are based on a 32-Bit RISC[i] CPU[ii] together with dedicated hardware blocks, including an error correcting code (ECC) unit, buffers, flash and host interface control logic.

The firmware, stored in the flash memory, is Flash, application, and host interface specific. All tasks with respect to flash, data management, and data transfers between flash and host are implemented either in hardware or in software. Hyperstone flash controllers boot-up using firmware that is stored within the flash memory of the product. Other solutions might store firmware in the ROM of the controller. Therefore, manufacturers are able to provide different products or feature sets all based on identical product hardware. Also, because the firmware is copied into the flash in a so-called 'pre-formatting' process after the storage product has been assembled, firmware can be updated in the field or immediately before delivery.

Flash Memory Management

Several algorithms and concepts are used to address the questions initially posed in re-writing to areas already containing data, maximizing flash life time, and ensuring data transfer integrity.

Blocks and Block Mapping

Logical block addresses (LBA) including the related static sector address or information are mapped correspondingly to physical block addresses (PBA). A table is maintained by the controller or firmware, translating requests for the particular LBA to its corresponding PBA. Logical blocks are distributed across all available flash chips and 'good' blocks, e.g. logical block 0 might correspond to a physical block at chip 1 block 2, and logical block 1 might correspond to a physical block on chip 3, block 3.

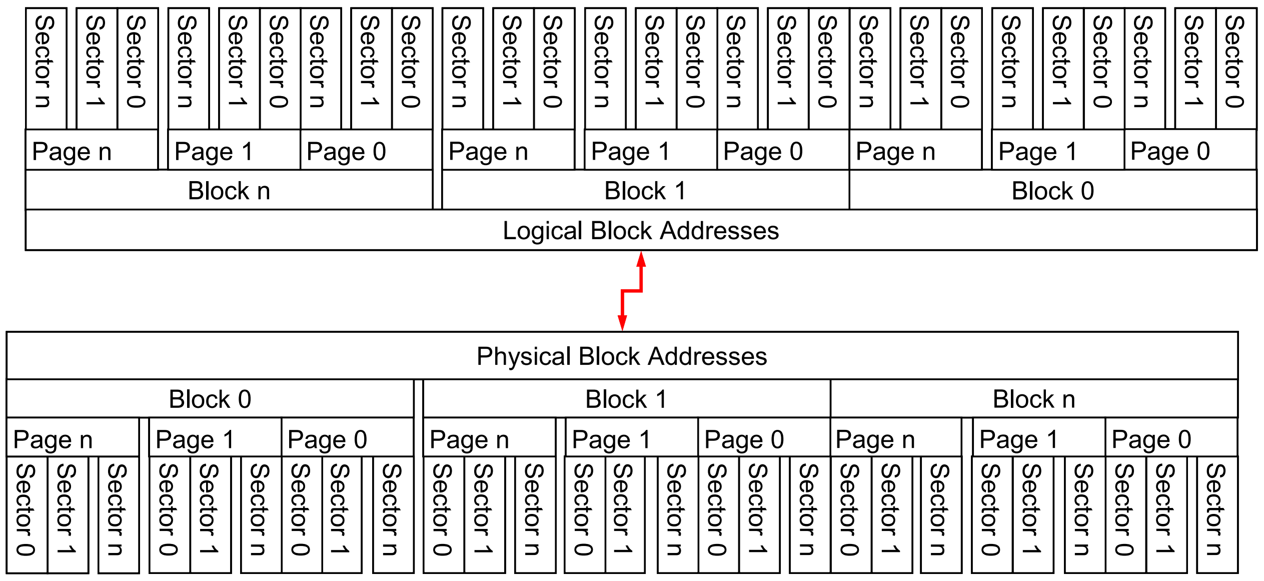


Figure 8: Logical to Physical Block Addressing

Management of the flash memory starts with the *anchor block*. This is used to store vital boot information, such as the location of firmware (boot sector), settings, configurations, and other static information that doesn't change. The so-called *commit block* contains permanent data, tables or references to tables and management blocks that are used for mapping, wear leveling, logbook etc. A pool of pre-erased *buffer blocks* is maintained, along with a table holding their status. *User blocks* are addressable blocks for user data. And finally, *defect blocks* that are unusable and a pool of *spare blocks* is then used to replace blocks that become bad during card lifetime.

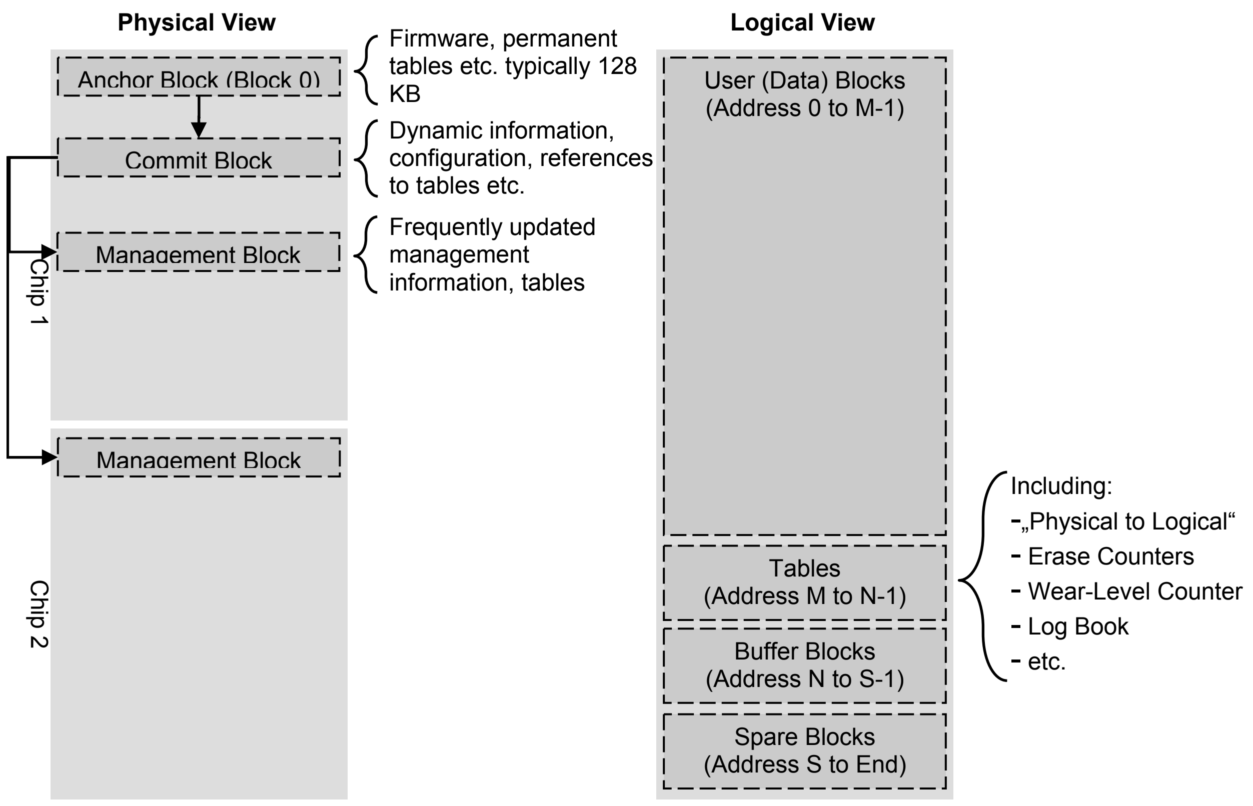

Physical View
Anchor Block (Block 0)
Firmware, permanent tables etc. typically 128 KB
Commit Block
Dynamic information, configuration, references to tables etc.
Management Block
Frequently updated management information, tables
Chip 1
Management Block
Chip 2
Logical View
User (Data) Blocks (Address 0 to M-1)
Tables (Address M to N-1)
Buffer Blocks (Address N to S-1)
Spare Blocks (Address S to End)
Including:
-„Physical to Logical“
- Erase Counters
- Wear-Level Counter
- Log Book
- etc.

Figure 9: Flash Management Physical versus Logical View

Error Code Correction (ECC)

The ECC is generally responsible for ensuring the quality of data being read, by adding information during the writing phase to restore partially corrupted data. ECC algorithms encode a larger data array based on certain assumptions or criteria in order to detect and repair errors. To get into the logic, one could start with a simple error detection method such as a parity bit, check sum or cyclic redundancy check (CRC). A check sum is as simple as adding all bits and comparing results. This is an easy and efficient method in terms of data overhead for error detection but doesn’t help to repair data. Therefore more complex encoding is needed. So called syndromes contain more information that not only detect but can also help to restore data. Encoding additional information requires more overhead and calculation power. The Hamming code, a linear error-correcting code based on a parity matrix, has been widely used in telecommunications.

Data Package	Data Bit A	Data Bit B	Data Bit C	Data Bit D	Parity Bit 1 C1=A+B+C	Parity Bit 2 C2=A+B+D	Parity Bit 3 C3=A+C+D
P1 sent	1	0	0	1	1	0	0
P2 sent	0	1	1	0	0	1	1

Figure 10: Example of a Hamming Code encoding

Data Package	Data Bit A	Data Bit B	Data Bit C	Data Bit D	C1’ = A+B+C	C2’ = A+B+D	C3’ = A+C+D	C1=C1’? C2=C2’? C3=C3’
P1 received	0	0	0	1	0	1	1	ERROR
P2 received	0	1	1	0	0	1	1	O.K.

Figure 11: Example of a Hamming Code used for Error detection

Identified False Bit	C1’ = A+B+C	C2’ = A+B+D	C3’ = A+C+D
None	True	True	True
Data Bit A	False	False	False
Data Bit B	False	False	True
Data Bit C	False	True	False
Data Bit D	True	False	False
Parity Bit C1’	False	True	True
Parity Bit C2’	True	False	True
Parity Bit C3’	True	True	False

Figure 12: Table of Equations and results for all possible error bits

By performing the following calculations and comparing the results to the above table, the false bit can be identified and corrected as necessary:

C1’= A XOR B XOR C = 1 -> False since the value of C1’ received is 0)
C2’= A XOR B XOR D = 0 -> (False since the value of C2’ received is 1)
C3’= A XOR C XOR D = 0 -> (False since the value of C3’ received is 1)

As all values are false, Data Bit A was received wrong and can now be corrected. This example shows how an ECC algorithm can work and also is an example as to the overhead required in terms of additional data that needs to be stored and transmitted.

As for the Hamming Code, 3 bits had to be added to 4 bits of information. This ratio might not be considered very efficient. Reed-Solomon code works similarly for multi-bit symbols. A (255,235) Reed-Solomon code RS (255,235) - specifies a total block length of 255 bytes (or symbols); 235 bytes used for information and 20 check bytes. The check bytes are calculated in a similar manner to the 3 check bits in the Hamming code example above and are appended to the end of the data block. Reed-Solomon codes are much more complex however, and require a significant amount of arithmetic and logical processing. Reed-Solomon codes are very code rate efficient. Within Flash handling, error correction is done on a bit/byte level within sectors, i.e. a 4-Bit Reed-Solomon ECC, refers to the capability of correcting 4 bytes, that is a maximum of 32 Bit, within a sector of 512 (+16) data byte.

Therefore, Flashes specify a so-called spare area for each page, for example when we talked about a 512 byte or 4Kbyte page before, we actually refer to e.g. 528 and 4224 Bytes per page where the additional memory is not available to users but used by an ECC to store overhead information. This also results in the fact that the ECC can be limited by the spare area of each page provided on the flash.

Both Reed-Solomon and the Hamming Code are good for burst errors as are more common for data transmission and telecoms. Since Flash cell errors occur more sporadically other algorithms are appearing in products, such as a more flexible binary BCH[iii], for which parity and syndrome calculation as well as correction can all be done in hardware. For this concept, six bits are correctable for a 16 byte overhead area per 512 data bytes, or 14 bits correctable for a 32 byte (30 byte or larger) overhead area. In general, since ECC algorithms can either be implemented in software or in hardware there is a trade-off between flexibility to support the many different flash memory cell architectures and performance of the ECC. Because of the very small cell structures in new MLC Flash, ECC tasks are of increasing importance and also increasingly specific to cells’ architectures.

The Hyperstone F3 controller for SSD/IDE/CF applications uses a 4 symbol Reed-Solomon ECC. Parity and syndrome generation are performed on-the-fly (in hardware) during the data transfer to/from the flash. Correction itself is performed in software, with Galois field arithmetic support through special CPU instructions. Additionally and in order to make sure the ECC has worked correctly, a cyclic redundancy check (CRC) is added in order to check if the resulting "corrected data" is indeed correct. Correctable errors are corrected without further notice. Uncorrectable errors in user data are notified to the host setting an "UNC bit" in the error register. The Hyperstone S7, a dual-controller for SDHC memory cards and MMC, uses a 6/14bit ECC based on BCH algorithms.

Bad Block Management

In order to ensure that all blocks used for data storage are functional, but knowing that blocks wear out or become defective for several reasons, bad block management is necessary to retire all questionable blocks. During initial pre-formatting, flash memories are tested and bad blocks originally marked by the Flash manufacturers are subsequently mapped out. Bad blocks are entered into a bad block table referring to physical block addresses. A pool of spare blocks is defined and used for dynamic bad block replacement, where blocks from the pool are used to replace blocks that produce errors when erased or programmed. This is called 'bad block re-mapping'. With Hyperstone controllers, the size of the spare block pool, while usually in the range of 2%, can be customized and has some impact on product lifetime. Considerations as to when blocks should be considered "bad" range from recognizing write or erase fails, reported by the flash itself, to algorithms based on bit errors recognized by the ECC during reading. During the flash's life within an application and while managed by an external controller, the bad block table is maintained and extended by any block going bad as it is mapped out or 'retired'. As soon as all spare blocks are consumed, Hyperstone controllers can optionally set the drive or card into read only mode.

The probability of bit errors, or the bit error rate, increases with the number of write/erase cycles, with smaller process geometries, and also for MLC compared to SLC architectures. Therefore, a more powerful ECC (e.g. other than specified in the flash data sheet) can extend the lifetime of blocks over the specified number of write/erase cycles. Currently, the ECC is implemented as part of the controller and is responsible for the quality of reading from flash. The quality of writing and erasing is determined by circuits within a flash. This is important, as both *do not* necessarily communicate. Therefore, it is possible that the flash reports a write/erase failure too early, when indeed the external ECC would be capable of correcting the errors. On the other hand, the controller on the flash does not consider information regarding the bit error rate of reading. Finally, bit errors when reading do not necessarily mean a block is indeed bad, simply that the information is corrupted to some extent. Re-writing the information into another block might cure the system altogether if in fact the data corruption was due a temporary external effect and not to a worn out block. Erasing that involves writing the value "1" for SLC or "11" for MLC is of great significance in managing the flash. It is the most strenuous of activities for the gate hence the one where most errors occur. At the same time it is the easiest to verify since the result for each cell is known and can be easily checked, such as building a check sum for instance. Therefore, pre-erasing can function almost like a gate keeper, weeding out bad blocks, if the process is managed in the right way.

Wear Leveling

Hyperstone's firmware provides patented algorithms balancing the use of all blocks, thus guarantying maximum lifetime of products. Furthermore, algorithms relating to and integrated in the wear leveling can help to avoid capacitive or parasitic-coupling, floating-gate-to-floating-gate noise, electron contamination and such by optimizing the spread of cells being addressed. Hyperstone continually evolves wear leveling algorithms to address ever changing Flash architectures and smaller process geometries.

Hyperstone's patented wear leveling algorithms are erase count triggered. All blocks are classified into so-called 'wear level classes'. Whenever a block is erased, a counter is increased. If the erase counter reaches a defined threshold, the block's wear level class is increased. By comparing wear level classes of blocks to be used, the load is balanced throughout all blocks on a flash memory chip. By trying to ensure that as few blocks as possible reach their end-of-life before others, and together with defining an adequate size of the spare block pool, the products life can be maximized and data integrity be ensured.

Hyperstone provides vendor specific commands, providing, amongst other features, information regarding wear-level classes of blocks. Using the command “read wear level count”, system developers can read block usage statistics. This information combined with defined thresholds can serve as an endurance status indicator.

Endurance

Floating gates “wear out” and after a certain amount of program/erase cycles, a gate becomes defective. A certain amount of defective gates can be compensated by the ECC. As soon as the amount of defective gates (bit error rate) within a block is larger than the ECC can handle, a block is mapped out as a bad block.

Usually, endurance of SLC Flash is specified with 100K program/erase cycles, while endurance of MLC flashes reach up to 10K program/erase cycles. Typically a reference to some kind of ECC is specified along with it, which could be 1-Bit for SLC and 4-Bit for MLC.

The following calculation only holds true under the assumption that perfect wear leveling is applied and all static data that has been programmed is still involved in the pool of blocks. As a very first approximation, the total “cycle potential” could be estimated by multiplying the number of blocks by the specified maximum program/erase cycles. For example, with 8K blocks and 100K program/erase cycles, we could assume a “cycle potential” of a single SLC flash to be 800 million [block write/erases].

In actuality, estimating a products’ lifetime is more complicated because one would need to know how this “cycle potential” is “consumed”. Let’s look at a scenario where we would re-write a file equal to a block size of 256 kB. With 8.000 blocks we could write 100.000 times to each block that is 800 million write erase cycles in total. Physically that file would have been located 100K times in each block.

On the other hand, looking at a scenario of rewriting a file that is equal in size to the total available memory, in our example that is 2 GByte, wear leveling or page wise programming would not have any effect. This file could be written 100K times before reaching the specified limit.

In order to translate a “cycle potential” into a lifetime in years it must be known how frequently data is written. Again, taking a very theoretical example of a block size file and assuming programming happens every second, the 800 million cycles would result in a life expectancy of about 25 years (800 million / 1x60x60x24x365). For MLC this would still be 3 years of continuously writing that file each second. The administrative overhead in terms of wear leveling etc. for Hyperstone controllers is only in the range of 1 -2%. The following tables can give a rough estimate of the order of magnitude regarding this endurance approximation.

File Size equal to:	e.g.		2 GB, SLC Flash	2 GB, MLC Flash	
One Block	256	kB	800,000,000	80,000,000	# cycles
8 Blocks	2	MB	100,000,000	10,000,000	# cycles
80 Blocks	20	MB	10,000,000	1,000,000	# cycles
800 Blocks	204.8	MB	1,000,000	100,000	# cycles
Capacity	2,048	MB	100,000	10,000	# cycles

Figure 13: Example of an endurance approximation in number of write/erase cycles

File Size equal to:	e.g.	write frequency	2 GB, SLC Flash	2 GB, MLC Flash	
One Block	256 kB	1 x each second	25	3	Years
8 Blocks	2.048 MB	1 x each minute	190	19	Years
80 Blocks	20.48 MB	1 x each minute	19	2	Years
800 Blocks	204.8 MB	1 x each hour	114	11	Years
Capacity	2,048 MB	1 x each hour	11	1	Years

Figure 14: Example of an endurance approximation in years based on assumed data write frequencies

Without wear leveling, endurance estimation heavily depends on what address might be written to. Even a very small file, if written to the same address over and over again, might destroy blocks fast and in the extreme worst case the overall endurance is very close to the specified write/erase cycles.

Another important factor to endurance is the quality of the ECC compared to the statistical occurrence of read errors. Usually, a kind of ECC is specified along with the number of write/erase cycles e.g. 1-Bit for SLC and 4-Bit for MLC. Using a better ECC can increase these numbers significantly.

Safe Power Loss Protection

Flash memory is often used in removable storage applications or battery operated devices where a robust and reliable power source cannot be guaranteed. A user may remove the memory at any time and under these conditions security of data is of paramount importance. Hyperstone has developed a patented concept in order to ensure data integrity when transferring or writing data. By using certain buffer blocks, information is written in a way that minimizes the delta between an old and a new state. The data system is coherent at all times.
Upon a sudden power fail, the controller is reset and the flash is immediately write-protected. A log of the most recent Flash transactions is kept, where entries are made just before any programming to the Flash. Should the last entry of the log be corrupted, the controller recovers the last valid entry. This minimizes data loss due to power failures and data corruption at the physical layer is prevented completely. Should power loss happen at the very same time when data is written to the flash, this data might get lost. In no case, however, will the overall data system be corrupted. Hyperstone performs extensive power cycling tests to all controllers and firmware verifying no data corruption due to power failure.

Interleaving, Multi Plane Operations, and Copy Back

If two flash chips are connected, one to each of two channels, interleaving operations are possible. Interleaving basically doubles the performance minus some negligible arbitration overhead. For some flashes, blocks have also been grouped into several planes. For example, a 16 Gbit flash might be organized in 4 planes containing 2,048 blocks each and allow simultaneous page programming or block erasing by selecting a page or block from each plane. The block address map can be configured so that two-plane programming or reading can be performed to blocks within planes 0 (2) and 1 (3) simultaneously. In effect, two-plane programming is increasing the performance similar to interleaving. Unlike interleaving it does not require a two channel approach of the controller and flash interface. Instead the two planes are addressed within one flash chip.

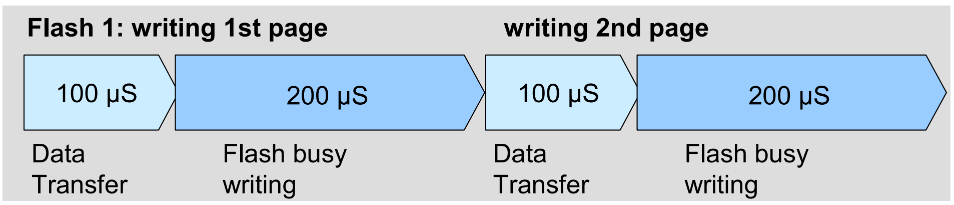


Figure 15: Example of writing to an SLC Flash with single channel configuration and without 2 plane operations where writing takes about 600 µs

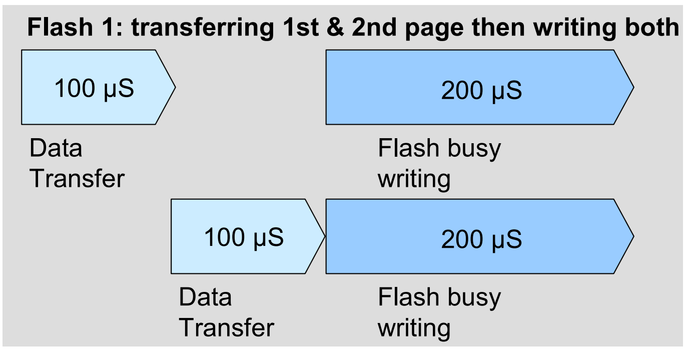


Figure 16: Example of writing to an SLC Flash with single channel configuration and with 2 plane operations where writing takes about 400 µs

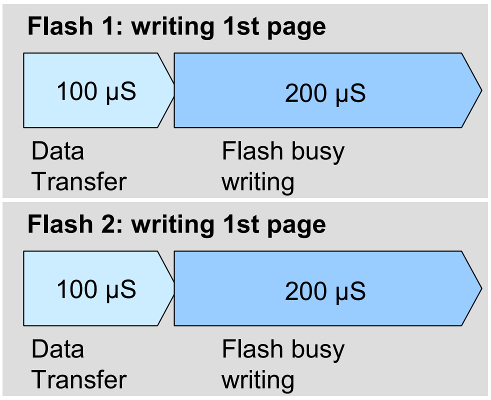


Figure 17: Example of writing to an SLC Flash with 2-channel configuration and without 2 plane operations where writing takes about 300 µs

The next figure compares the theoretical performance maximum of writing and reading for different Flash types and configurations with and without interleaving:

								Sustained Performance Calculated* [MB/s]							
								Read				Write			
Example data from selected datasheets								1 Channel		2 Channel		1 Channel		2 Channel	
Device Type	Cycle Time [ns/ byte]	Busy Read [µs/ page]	Busy Write [µs/ page]	Page Size [Kbyte]	Block Size [pages]	Block Erase time [ms]	Transfer Time [µs/page]	w/o 2 Plane	with 2 Plane	w/o 2 Plane	with 2 Plane	w/o 2 Plane	with 2 Plane	w/o 2 Plane	with 2 Plane
Flash #1 SLC	25	25	200	4	128	1.5	102.4	31	35	63	70	13	19	25	37
Flash #2 SLC	25	30	200	2	64	2.0	51.2	25	30	49	60	7	11	14	22
Flash #3 MLC	30	60	800	2	128	1.5	61.4	16	22	33	44	2	4	5	8
Flash #4 MLC	25	60	800	4	128	1.5	102.4	25	30	49	60	4	8	9	16
Flash #5 MLC	25	60	800	2	128	2.5	51.2	18	25	36	49	2	4	5	8

Figure 18: Example performance comparison between 2 channels interleaving and 2 plane operations

Yet another feature of flash is the so called 'copy back' operation, where a page is copied directly within the flash eliminating the time needed for data transfer. While performance can be increased to some extent, the data is not handled by the controller's ECC and read errors might not be properly corrected.

Hyperstone F3 Hardware Architecture

As an example of a more complex controller, the Hyperstone F3 offers two channels, "Enhanced Direct Flash Access" each with a 4 symbol Error Correcting Code (ECC) unit capable of correcting 4 Bytes in a 512 bytes sector. Up to 16 flash memory chips (eight chip-selects per channel) can be directly connected. The Hyperstone 32-Bit RISC microprocessor can be scaled, running at clock frequencies from 10 MHz to 60 MHz using a trimmable internal oscillator. The F3 offers 16 Kbytes internal Boot ROM, 20 Kbytes internal SRAM, four 512 Byte sector buffers and 256 Byte PCMCIA attribute memory. Performance with SLC flash memories above 40 MBytes/s reading and over 30 MBytes/s writing is achievable. With automatic power down mode during wait periods for host data or flash memory operation completion and automatic sleep mode during host inactivity periods, an Icc[iv] of less than 200 µA can be achieved.

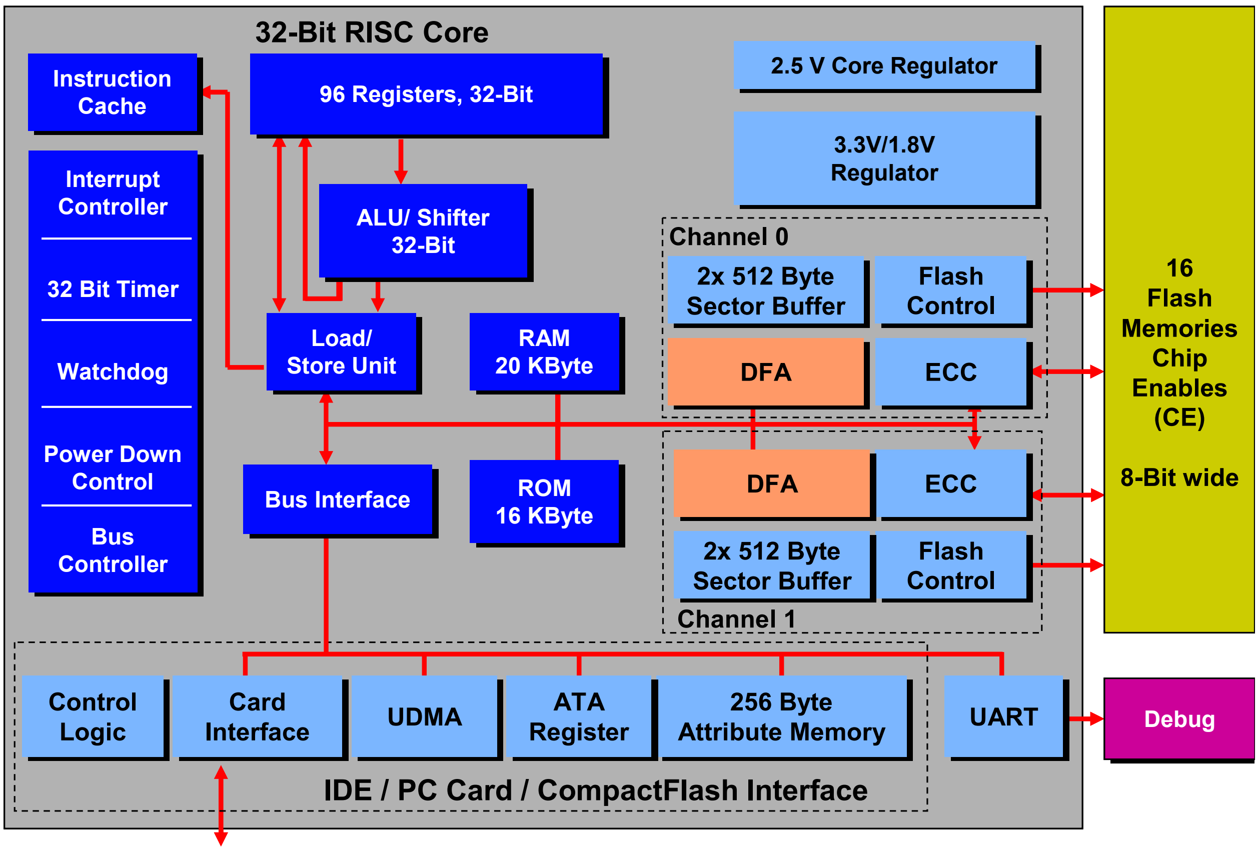


Figure 19: Block Diagram of Hyperstone F3 Flash Memory Controller for CF cards or Solid State Disks

The F3 is specifically designed for applications such as high-speed CF cards, Disk-on-Module, or Solid State Disks. It supports automatic sensing of PCMCIA[v] or True-IDE[vi] host interface mode, and compliant to PCMCIA 2.1, PC Card ATA, CF 3.0, memory mapped or I/O operations, fast ATA host-to-buffer transfer rates, PIO[vii] mode 6, MDMA mode 4, and UDMA[viii] mode 4 in true-IDE mode. Serial-ATA (SATA) can also be realized using an additional PATA to SATA bridge chip.

Hyperstone S6 Hardware Architecture

Addressing different applications the Hyperstone S6 controller offers two channels, “Enhanced Direct Flash Access” each with a 4 symbol Error Correcting Code (ECC) unit capable of correcting 4 Bytes in a 512 bytes sector. Up to 4 flash memory chips (two chip-selects per channel) can be connected directly. The Hyperstone 32-Bit RISC microprocessor can be scaled running at clock frequencies from 10 MHz to 60 MHz using a trimmable internal oscillator. The S6 offers 16 Kbytes internal Boot ROM, 20 Kbytes internal SRAM, and four 4KByte sector buffers optimized for MLC with 4kB page size. Implementing the high-speed or SDHC SD2.0 and MMC 4.2 specifications, performance in SD mode up to 24 MB/s sustained read, 23MB/s sustained write for SLC and 22MB/s sustained read, 9.2 MB/s sustained write for MLC is achievable. In MMC mode sustained read speed over 40 MB/s is realistic. Dual-voltage support for 1.8V and 3.3V flashes including the ability to support 3.3V flashes when connected to a 1.8V host and an optimized pad layout make the S6 suitable for microSD and eMMC. The S6 will soon be complemented by the S7, adding a more advanced 6 / 14-Bit BCH ECC plus 4 additional chip enables.

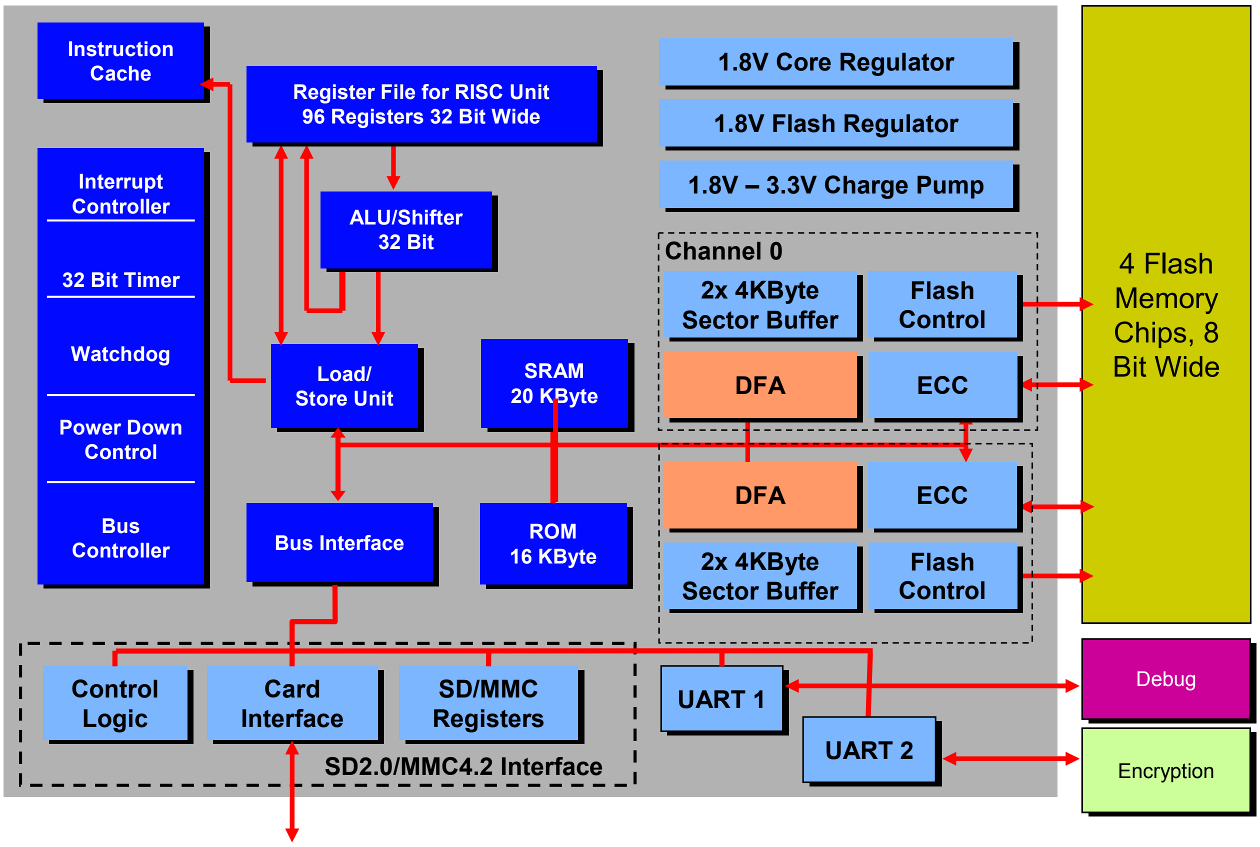


Figure 20: Block Diagram of Hyperstone S6 Flash Memory Controller for SD, microSD Cards & eMMC embedded Flash

Solid State Disks (SSD) versus Hard Disk Drives (HDD)

Following flash cards and removable data storage systems, the latest killer application for flash is considered to be solid state disks (SSD), a substitute for hard disk drives or fixed storage devices. Flash has some significant advantages over hard disk drives most of which due to the absence of moving parts, power consumption and robustness to physical impact:

- Performance: Latency times basically due to the inherent mechanical operations of HDD are not necessary for SSD and seek times or access times are about 30 times faster. Sustained writing using interleaving can be scaled to some extend by both HDD and SSD. However, again due to the mechanical issues, this is more difficult for HDDs. Finally the HDD has a slight advantage at sustained reading.
  Performance in real life scenarios also depends on the level of fragmentation of the data when using HDDs. Have you used your defrag tool lately? Or have you ever performed this for a 500GB drive? With SSDs, defragmentation is an inherent feature of the logical to physical translation and wear-leveling.
- Cost: storage capacity independent bottom line cost determined by BOM is about 50$. Therefore, HDD is cost competitive for large capacity but not for small capacity drives.
- Power Consumption:
  - Power Idle: HDD 0.8–5.0 W; SSD 0. 035–3.0 W
  - Power Read/Write: HDD 5.0–10.0 W; SSD: 0. 325–5.0 W
- Reliability: Again due to the mechanics HDD most significant disadvantage is its reliability and sensitivity due to shock
- Scalability of Form Factors: Solid State Storage can be implemented in many different form factors from microSD or eMMC to 3.5” drives.

- Choosing the optimal Host Interfaces: Solid State Storage can be based on several host interfaces including but not limited to SD, MMC, USB, S-ATA, P-ATA/IDE, CF, while HDDs are mainly driven by the PC market using mainly S-ATA today.
- Endurance (Write/Erase Cycles) HDD: NA, SSD: 300,000 - >5 million

Feature Possibilities & Emerging Applications

Several features can be realized aside of standard features. Cards can be defined as read-only, content can be protected, and writing to a card can be limited or defined in number, e.g. write-once, -twice, x times. Event triggered write-protection is possible, as is making cards function only in certain devices. Proprietary hardware format can make use of standards such as SD/MMC, ATA, or IDE together with firmware comprising a disk-on-board. Requirements for qualification, tight quality control, or simply longer life cycles compared to consumer products might easily be addressed, and result in specific products or in-sourcing.

Requirements are changing

New generations of flash will continue to improve performance, increase capacity, or drive cost per MByte down. These advances include technology shrinks (60, 50, 40 nm), larger page sizes (2KB, 4KB), several flash interfaces (8-Bit, 16-Bit, ONFI, HL-NAND), and finally cell technologies with different write or erase cycle performance and maximum life time (SLC, 2-Bit MLC, 4-Bit MLC). Furthermore, new packaging and manufacturing processes will continue to drive overall system cost down and capacity up as well. All of these advances have an impact on the controller or firmware implementation and as a result, controllers and firmware features will become more important as an enabling and competitive factor of applications.

While ensuring highest performance, reliability, enabling highest capacity, guaranteeing data integrity and maximum life-time are most important features determined by controllers, cost of controllers together with firmware represent only a small fraction of the overall systems' cost.

Conclusion

The cost of an overall flash based storage system is mainly driven by the flash memory component. Decreasing cost of flashes results in more complex tasks for the systems' flash controllers and a fine-tuned balance between hardware and software is necessary for these controllers to provide a robust and high performing system.

Controllers and their firmware are the heart of flash storage systems. Features and performance in the controller are vital to the viability of a Flash storage application. Hyperstone controllers have become the choice of leading companies around the world by providing excellent read and write performance,, ensuring highest reliability and data integrity, supporting large attachable storage capacity, and extending storage media lifetime .

---

[i] RISC: Reduced Instruction Set Computer
[ii] CPU: Central Processing Unit
[iii] BCH: Bose, Ray-Chaudhuri, Hocquenghem code

[iv] Icc: current (I) supplied to the collectors of transistors in a circuit
[v] PCMCIA: Personal Computer Memory Card International Association
[vi] ATA: Advance Technology Attachment, a standard interface for connecting storage devices; synonyms include Integrated Drive Electronics (IDE), AT Attachment Packet Interface (ATAPI), and Parallel-ATA (PATA).
[vii] PIO: programmed input/output (PIO)
[viii] DMA, UDMA: Direct Memory Access, Ultra DMA