

A High Performance Controller for NAND Flash-based Solid State Disk (NSSD)

Chanik Park, Prakash Talawar, Daesik Won, MyungJin Jung, JungBeen Im, Suksan Kim and Youngjoon Choi

Samsung Electronics Co., Ltd., San #16 Banwol-Ri, Taejeon-Eup, Hwasung-City, Gyeonggi-Do, Korea

Abstract

NAND Flash memory based solid-state disk (NSSD) has been used for industrial and military use due to its high reliability and shock resistance. With the bit cost reduction of flash memory and the explosive growth of flash market, NSSD is expected to penetrate into diverse applications such as mobile thin clients, car navigation systems and movie players, which prefer low power consumption, high reliability, high performance and so on. This paper will mainly focus on the development of a high performance and cost-efficient controller for NSSD, with the aim of describing both hardware and software architectures. In order to demonstrate the usefulness of the proposed approach, we will show performance, power consumption and start-up time evaluation results over magnetic disks using third party benchmark tools.

Introduction

The main driving forces for reducing the bit cost of NAND flash memory are MLC (multi-level cell) technology and migration to lower geometries (70nm and 55nm). The combination of two factors will enable 40-50% cost reduction year over year as shown in Figure 1.

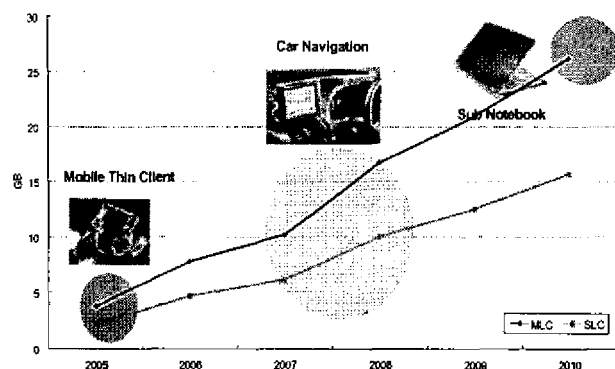


Figure 1. GB Flash for the price of US\$120

The proliferation of a wide range of mobile applications from consumer and automotive to computer and communication high necessitates the following requirements in addition to high density.

- Fast start-up time
- Cost efficiency
- High performance read/write
- Low-power consumption
- Reliability

Although several commercial products of NSSD can be found in [1][2][3] satisfying some of the above requirements, technical discussion on NSSD controller can be hardly found in the prior literature.

In this paper, we present hardware and software architecture for NAND Flash based massive storage device, namely NSSD.

Hardware architecture

Previous approaches for NSSD have been based on multi-chip solutions in order to obtain the extreme performance of multiple flash memories as shown in Figure 2. In particular, the use of DRAM as code and data memory may result in cost inefficiency as well as high power consumption, which is not suitable for the portable storage devices. In case of the rotating magnetic disk such as the hard disk drive, DRAM can be utilized as a write buffer for the purpose of hiding the seek time of the rotating disk during the host data loading. However, as for NSSD, NAND Flash memory itself can be used as a write buffer due to its random accessibility without seek overhead and fast erase/program performance.

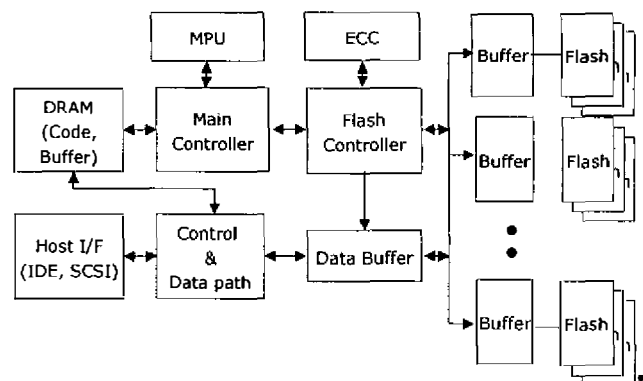


Figure 2. Existing multi chip architecture for NSSD

The proposed NSSD hardware architecture is shown in Figure 3. As NSSD can be interfaced directly with the Host using the IDE standard interface (ATA-5), it supports up to Ultra DMA Mode 4. As far as flash interface is concerned, the controller can support 2 to 16 pieces of 1 to 16 Giga bit NAND flash memories [5]. As a result, the total capacity can range from 256MBytes to 32GBytes.

The controller supports 2 channels and up to 4-way interleave to hide Flash program latency and to increase write throughput. The flash memory chips are interfaced to each channel to support parallel write operations.

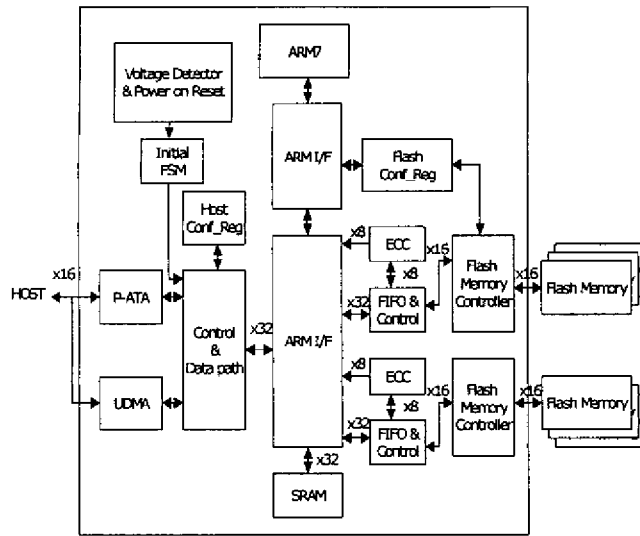


Figure 3. The proposed hardware architecture

According to the analysis of host system's access pattern, sequential write is dominant in normal user applications as well as multimedia applications. Therefore, the key factor of improving NSSD performance is to accelerate sequential write operation using interleaving scheme of multiple NAND flash memories. We implemented automatic interleaving hardware logic to maximize the interleaving effect, while the firmware intervention is minimized as shown in Figure 4.

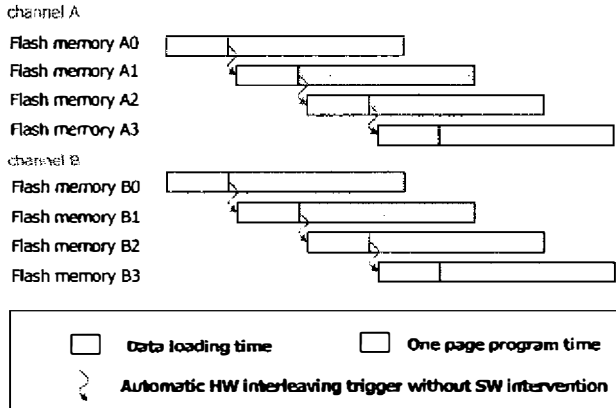


Figure 4. Hardware driven 2-channel 4-way interleaving without firmware intervention

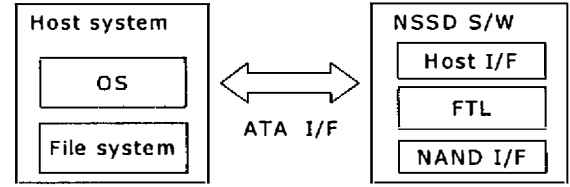
In order to reduce power consumption, the controller uses a small size of SRAM for program¹, data and buffer memory instead of DRAM. The controller also supports all the power modes like active, idle, standby and sleep mode. The controller's implemented chip size is 5mm x 5mm with 0.18μm process. Compared with multi-chip solution for SSD, our on chip solution can be applied to various applica-

¹ At power up time, the controller automatically copies software code from the fixed location of NAND flash to SRAM.

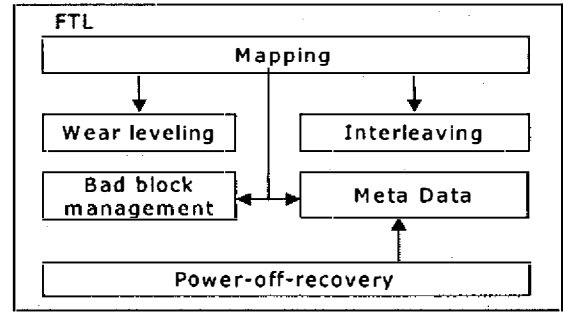
tions ranging from consumer electronics to industrial applications due to its cost efficiency and low power advantages.

Software architecture

The high level architecture for NSSD software is shown in Figure 5.



(a) Host and NSSD S/W structure



(b) NSSD FTL structure

Figure 5. The proposed software architecture

NSSD software consists of three layers:

- 1) Host I/F layer: this layer is responsible for interpreting the host commands and invoking the associated FTL functions such as read and write operations.
- 2) Flash translation layer (FTL): FTL is the main part of the NSSD software which converts logical address from the host into the physical address of the flash memory and it has many other algorithms to fully exploit the architectural features of NAND flash memory, which are explained below in brief.
- 3) NAND I/F layer: this layer is the low level driver to access the flash memories through NAND Flash controller.

Hybrid Mapping

The mapping between the logical address and the physical address can be handled either at the page level or at the block level. Page mapping has advantage of flexible management of the write data location in a block (out-of-place mapping). On the contrary, in case of block mapping, the write data should be written to the fixed location in a block (in-place mapping). In view of memory requirement, page mapping consumes more memory than block mapping due to its page map table size. In our implementation with limited SRAM size, the compromised solution of page mapping and block mapping is adopted, namely hybrid mapping. The detailed explanation about hybrid mapping mechanism can be found in [4]. Super block (a fixed number of physical blocks are grouped to make one super block) scheme is used

not only to reduce the block mapping table size, but also to make use of interleaving scheme. Hybrid mapping uses the *write buffer* block (log block) and data block concepts to efficiently handle frequently updated data and sequentially appended data. Write buffer (Log block) block contains the out-of-place pages and data block contains the in-place pages.

When the FTL receives the write command, it allocates one log block and starts writing into that block. Since the data is out of order in log block, a page map table for each log block is provided. The page map table is updated for each write command. If all the physical pages of the log block are full, merge² operation is performed. Merge operation is costly operation since it involves as many page copy operations as the maximum number of pages of a block (i.e., 64 pages) in the worst case. Therefore, log block management should be carefully handled. In our implementation, least recently used (LRU) method (i.e. the log block which is not used recently among the log blocks would be selected for merge operation) is used for log block merging operation. This policy turned out to be beneficial in improving the performance for frequently updated writes. In particular, the number of log blocks has strong impact on performance since the small writes at the random locations can be buffered in the log blocks. However, the number of log blocks is limited due to the SRAM size.

Wear leveling

The circular queue table is maintained in the software which consists of both erased and dirty (not erased) blocks. Dynamic wear leveling is achieved by using the circular queue table for Garbage (free) blocks. During the write operation the log blocks are allocated from the head of the queue table and the dirty blocks occurred during merge will be inserted into the tail of the table. The overall block map table would be modified. If there is no free block, dirty blocks will be erased from the circular queue table. During the run time the meta data blocks and data blocks are exchanged with the garbage blocks. Circular queue table provides implicit wear leveling by circulating the blocks in the queue. The dynamic wear leveling guarantees that all flash blocks will be erased approximately same number of times. As a result, the life time of the product can be lengthened up to the maximum endurance of the product.

Bad block management

There are two types of bad blocks 1) Initial bad blocks: these blocks are mapped out during the factory initialization. 2) Runtime time bad blocks: these blocks will be mapped out in the circular queue table during the run time. During the erase if it finds some block as bad block then this block would be removed from the circular table.

² When data is written to NAND flash, a free (erased) block is allocated for storing the coming data. When the allocated block gets filled, the valid sectors should be moved to another free block.

Power-off-recovery

It is important to rectify "in-between" states of data caused by a power failure when only part of the data was transferred written to the flash during the write operation until a power failure occurred but the flash meta data (sector/block mapping tables) indicates a different scenario. To avoid this situation meta data (i.e. sector / block mapping tables, pointer tables etc) is stored separately in meta data blocks. During the write operation the meta data is updated consistently so that the previous state meta data could be recovered completely during sudden power-off.

Performance results

We have used several industry standard third party benchmark tools to compare the performance of NSSD over commercial HDD [6]. The results measured in Samsung (Q-30) notebook are discussed here.

The "PC-mark" results show that NSSD performance is superior to HDD by 77% (Table 1). PC-mark benchmark evaluates the storage performance with four categories such as startup, application loading, file copying and general usage. According to our analysis, the number of log blocks strongly affects the PC-mark score since its patterns consist of small size of writes at the random locations. In other words, the more log blocks are used, the higher PC-mark score is obtained.

NSSD shows outstanding booting time (Table 2) compared with HDD since it can achieve fast read access time at the random locations during Operating System image loading. Even though scattered small size writes at the random locations occur, they can be buffered to log blocks based on hybrid mapping scheme.

According to the "Sandra" benchmark results, NSSD has higher performance especially in case of sequential write/read, while Buffered read/write is less because the DRAM buffer in HDD is involved (Table 3).

The power consumed by NSSD is less in all the modes of operation; it can increase the battery life in case of the products using NSSD as a storage device (Table 4).

TABLE 1. "PC mark05" [7] results measured for Samsung Q 30 notebook

	Total Score	XP-startup (MB/S)	App-loading (MB/S)	File copy (MB/S)	General HDD usage (MB/S)
HDD	2969	5.267	4.193	24.846	3.334
NSSD	5280	9.595	10.548	18.440	6.468

TABLE 2. "Bootvis"[8] results measured for Q-30 note book

	Booting time
HDD	18 sec
NSSD	7 sec

TABLE 3. "Sandra" [9] results measured for Samsung Q-30 notebook

	Buffered read (MB/S)	Sequential read (MB/S)	Random read (MB/S)	Buffered write (MB/S)	Sequential write (MB/S)	Random write (MB/S)
HDD	83	21	17	35	21	15
NSSD	56	57	57	13	34	13

TABLE 4. Power consumption data measured by Oscilloscope

	Active	Idle	Standby	Sleep
HDD	360mA	83mA	38mA	19mA
NSSD	152mA	52mA	17mA	15mA

Conclusion

In this paper, we have presented a high performance and cost efficient controller for NSSD which provides instant-on boot, enhanced read/write performance, high reliability and low power consumption. We envision that NSSD will enlarge its application area covering from car navigation system to sub-notebook as the bit-cost decreases and user requirements for reliable and energy efficient storage devices are increased.

For future work, more efficient hardware and software architecture should be devised to incorporate MLC NAND Flash memory into NSSD since the MLC NAND is characterized by less endurance cycle and longer program time. The possibility of using FRAM or PRAM as a special purpose buffer in NSSD will be a challenging issue to reduce the FTL overhead of meta data write and power-off recovery management.

••

References

- [1] BitMICRO Networks, "E-Disk ATA:2.5" IDE/ATA Flash Disk," <http://www.bitmicro.com>
- [2] SiliconSystems Inc, "SiliconDrive™," <http://www.siliconsystems.com>
- [3] M-Systems Inc., "IDE/SATA/SCSI FFD," <http://www.m-sys.com>
- [4] J. Kim, J. M. Kim, S. H. Noh, S. L. Min, and Y. Cho, "A Space Efficient Flash Translation Layer for CompactFlash Systems," IEEE Transactions on Consumer Electronics, Vol. 48, No. 2, pp.365-375, 2002.
- [5] Samsung Electronics Co., Ltd., NAND Flash Memory Data Sheet, 2005.
- [6] Toshiba, "1.8inch HDD Specification" <http://sdd.toshiba.com>
- [7] FutureMark, "PCMark'05," <http://www.futuremark.com/products>
- [8] Microsoft, "BootVis," <http://www.microsoft.com/whdc/system/sysperf/fastboot/bootvis.mspix>
- [9] SiSoftware, "Sandra 2005," <http://www.sisoftware.net>