4

Block Cleaning Process in Flash Memory

Amir Rizaan Rahiman and Putra Sumari
Multimedia Research Group, School of Computer Sciences, University Sains Malaysia, Malaysia

1. Introduction

Flash memory is a non-volatile storage device that can retain its contents when the power is switched off. Generally, it is a form of an electrically erasable programmable read-only memory (EEPROM) that offers several excellent features such as less noise, solid-state reliability, lower power consumption, smaller size, light weight, and higher shock resistant [1 – 5]. Flash memory acts as a slim and compact storage device. It's main applications are such as compact flash (CF), secured digital (SD), and personal computer memory card international association (PCMCIA) cards, for storage and data transfer in most portable electronic gadgets such as mobile phones, digital cameras, personal digital assistants (PDAs), portable media players (PMPs), gobal positioning system receivers (GPS), just to name a few.

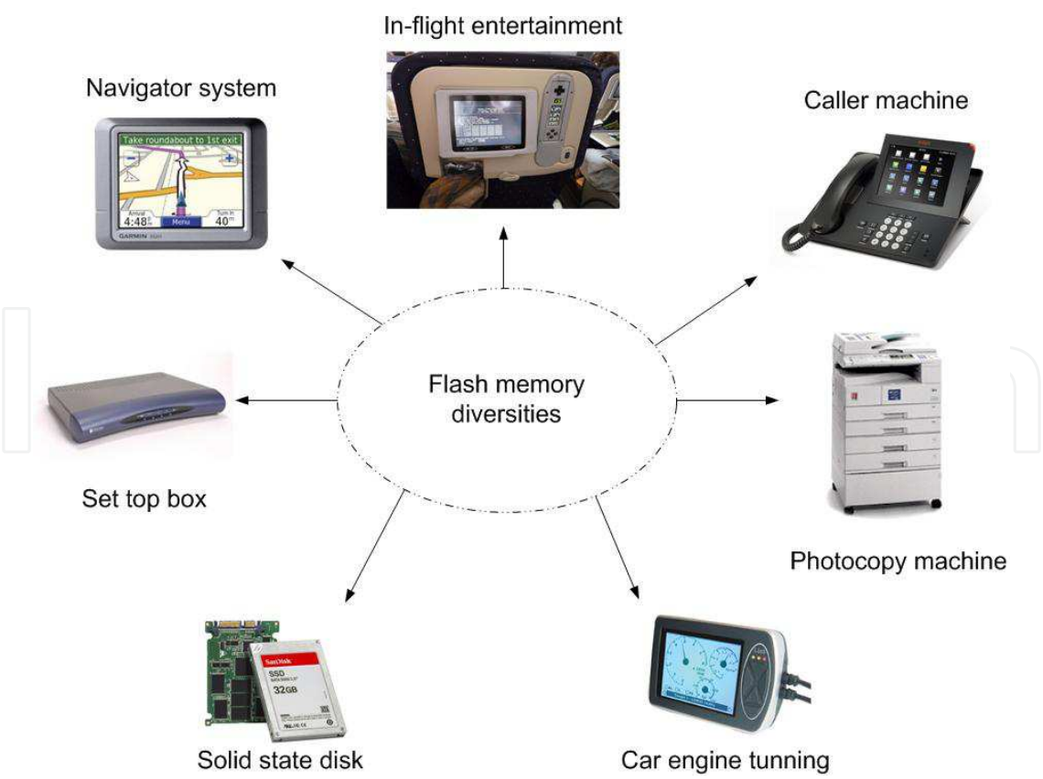


Fig. 1. Diverse applications of flash memory as embedded systems.

The demand for flash memory has reformed its usage to wide areas. For instance, as illustrated in Figure 1, flash memory is extensively used as embedded systems in several intelligent and novelty applications such as household appliances, telecommunication devices, computer applications, automotives and high technology machinery.

2. Flash memory architecture

As shown in Figure 2, flash memory is a block and page based storage device. The page unit is used to store data where a group of pages is referred to as a block. The page unit is partitioned into two areas, namely, 1) *Data* and 2) *Spare*. The data area is used to store the actual data while the spare area is used to store the supporting information for the data area (such as bad block identification, page and block data structures, error correction code (ECC), etc.). According to present production practices, the page size is fixed from 512 B to 4 KB, while the block size is between 4 KB and 128 KB [18]. Figure 3 shows the attributes of a 4 GB flash memory.

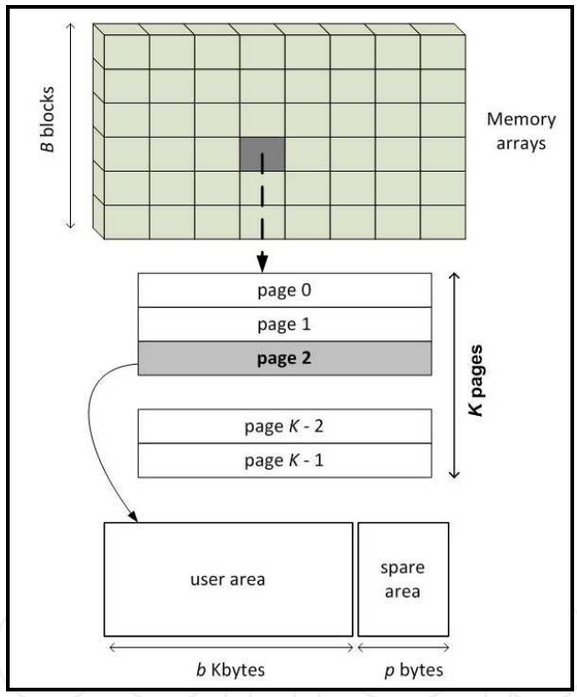


Fig. 2. Block and page layout in flash memory.

There are two different types of flash memory in the current market, namely, 1) *NOR-flash*, and 2) *NAND-flash* [2, 6]. The main distinction between both types is the I/O interface connection mechanism to the host system. The NOR-flash employs a memory mapped random access interface with a dedicated address and data lines that are similar to random access memory (RAM). Besides that, it is a byte-addressable data accessing device that permits random I/O access with higher performance in reading functionality. On the contrary, data access in NAND-flash is controlled and managed through two indirect I/O interface logic methods. They are the emulating block accessing method referred to as the flash translation layer (FTL) and native file system. The FTL allows physical accessing units

Symbol	Description	Value
-	Power consumption	2.70 – 3.60 V
-	Dimension (h x w x d) mm	12 x 20 x 1.2
R_t	Data read from a page into the data register	25 µs
W_t	Data write from a data register into memory page	200 µs
E_t	Block erasure access time	1.5 ms
S_t	Sequential access to the data register (bus data)	100 µs
V	Number of chips per device	2
B	Number of blocks per chip	8192
K	Number of pages per block	64
p	Page size	4 KB
b	Block size	256 KB
r_{data}	Data register	4 KB
D	Capacity per chip	2 GB
u_i	Block i utilization, $0 \leq i \leq B-1$	[0, 1]
-	Program/Erase cycles	1,000,000

Fig. 3. Flash memory attributes and specifications.

(block and page) to be addressed as a set of different accessing units (such as 512 B, 2 KB, 4 KB, depending on the manufacturers). In the native file system, the device accessing unit can directly be accessed without the translation layer. An example of the native file system employed in NAND-flash is the journaling flash file system (JFFS) [7] and yet another flash filing system (YAFFS) [23]. For application purposes, the NOR-flash is used for small amounts of code storage while the NAND-flash is mostly used in data storage applications since its characteristic are more similar to disk storage.

3. Flash memory characteristics

The characteristics of the flash memory can be summarized as follows [8, 9]:

i. **Free accessing time penalties**: The flash memory is a semiconductor device which eliminates the use of mechanical components. This allows the time required to access data to be uniform, regardless of the data's location. For instance, let's say both data *a* and data *b*, which are 4 KB in size each, are randomly located in block *i* and *k* (see Figure 4). The total time required to retrieve data *a* is 0.088 *ms* and data *b* is retrieved directly after retrieving data *a*.

   Data accessing (retrieving and storing) in flash memory is carried out in three phases, 1) *Setup*, 2) *Busy*, and 3) *Data transfer* [24]. The accessing command is initialized in the setup phase. In the busy phase, the required data is temporarily loaded into the flash memory I/O buffer within a fixed accessing time. Then, the stored data in the I/O buffer is transferred sequentially to the host system at every fixed serial access time during the data transfer phase. Similarly, the storing/writing process also requires constant access time, wherever the location might be.

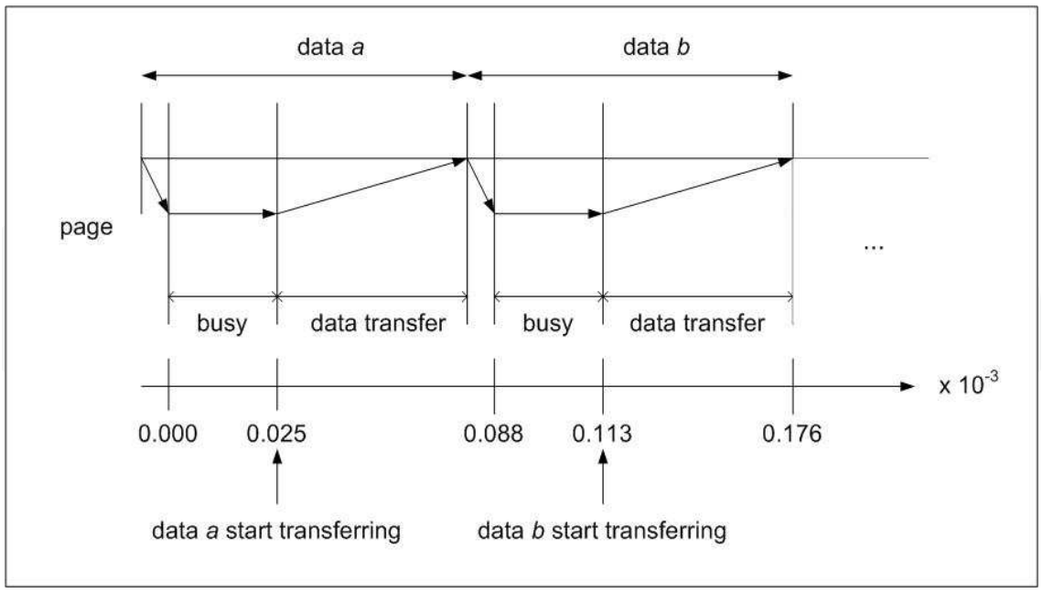


Fig. 4. Data accessing in flash memory array.

ii. **Out-place updating scheme**: Data updating in the flash memory is performed via an out-place scheme rather than an in-place scheme. Due to its EEPROM characteristic, if the in-place scheme is employed, the block where the updated data is located needs to be erased first before the data can be restored into the similar location. Furthermore, block erasure in flash memory is time consuming that can degrade I/O performance. Thus, the out-place scheme is employed where the updated data is stored into a new location while its original copy is set as garbage and will not be used any further [10 – 11] (see Figure 5). The main purpose of the out-place scheme is to avoid block erasure during every update process.

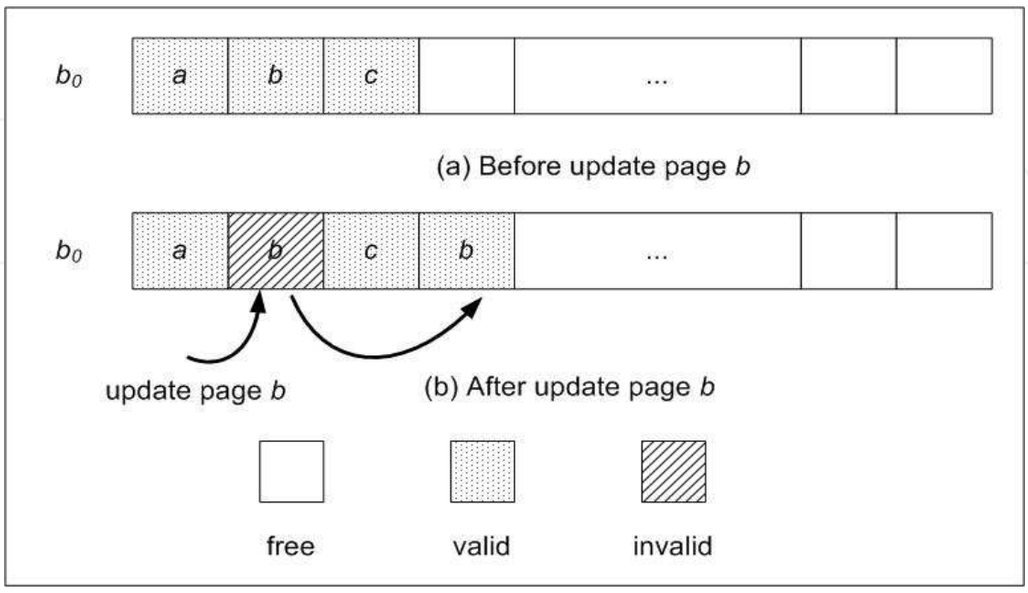


Fig. 5. The out-place updating scheme in the flash memory.

Due to the out-place scheme, the page in flash memory falls into three states, namely, 1) *Free*, 2) *Valid,* and 3) *Invalid*. The free state is when a page contains no data and is ready for storing/writing a new or updated data. The valid state is a page that contains the current version of the data while the invalid state refers to a page that contains garbage. In addition, the block status can either be active or inactive [12 – 13] due to the page states.

iii. **Asymmetric accessing time and unit**: There are three types of access functions in flash memory. They are 1) *Read*, 2) *Write/program*, and 3) *Erase*. Each function is realized in an asymmetric accessing unit and time (see Figure 3). The write function takes an order of magnitude longer than the read function and both are carried out in page unit, while the erase function requires the longest access time which is performed in block unit. The read function fetches a valid data from a valid target page, while the write function stores data (either new or updated) into a free target page. On the contrary, the erase function is used to erase an active or an inactive block with free or invalid pages.

iv. **Bulk cleaning with limited block life cycle**: The cleaning process is essential in flash memory due to the employed out-place updating scheme. The cleaning process is carried out on block unit rather than page unit. The block may contain valid data, thus, before initiating the process, all valid data residing in the block must be copied out into available free spaces in other free blocks. However, each block could be tolerant with a limited number of erasure cycles, for example one million (10^6) cycles. Exceeding the erasure cycles will cause the block to become unreliable and spoiled, permanently. For example, a multi-level cell (MLC) block type typically supports 10,000 erasure cycles. If the same block is erased and then re-programmed every second, the block would exceed the 10,000 cycle limit in just three hours. Thus, wear-leveling policy that wears down all memory blocks as evenly as possible is necessary [14, 15].

4. Cleaning process in flash memory

The cleaning process in flash memory refers to the process of collecting the garbage scattered throughout the memory array and then reclaiming them back into free space due to the out-place updating scheme. It is an essential process to guarantee free space availability on the memory array to ensure new data can be continually stored. However, the cleaning process is carried out by the erase function and involves bulk size of data rather than specific locations. The valid data in the block must be copied into the other blocks (free cells) first, before the cleaning can be initiated. Besides, each flash memory block could tolerate with an individual erasure lifetime. Frequently erasing blocks causes the blocks to become unreliable and thus, reduces physical device capacity.

The effectiveness of the cleaning process is heavily dependent on the efficient cleaning algorithm as well as the data allocation scheme employed by the flash memory system. Moreover, the cleaning process and the block utilization level are key to the cleaning process performance and have substantial impact on the access performance, energy consumption and block endurance [1, 10, 14]. Block utilization is the ratio between valid cells and total cells and it is represented in percentage. Two categories of cleaning processes in flash memory are 1) *Automatic*, and 2) *Semi-automatic* [16]. Both processes are initiated routinely by the flash memory controller.

4.1 Automatic cleaning

The automatic cleaning process is automatically commenced when a particular block's state in the memory array turns from an active to an inactive (all pages in the block have turned into invalid state or mixing with several number of free pages). Since there is no valid data copying process required, the block can be erased in the background during execution of the current I/O operations (such as read or write) from/into the memory array. Accordingly, this process requires a constant erase accessing time (E_t) where the target block ID is given to the memory controller to erase. Moreover, only a single inactive block (also known as victim block) can be erased each time the automatic cleaning process is commenced. In addition, the automatic cleaning process is influence by an efficient data allocation scheme engaged by the flash memory. There are several data allocation schemes in flash memory that share identical queuing techniques with a CPU scheduling policy such as first come first serve (FCFS), first re-arrival first serve (FRFS), online first re-arrival first serve (OFRFS), and Best Matching (BestM) [12 – 13]. Unlike CPU scheduling policies, the main objective of the data allocation scheme in the flash memory is to minimize the amount of active blocks required. The scheme requires the lowest amount of active blocks to minimize the amount of blocks to be erased when the actual cleaning process is initiated due to the limitation of the out-place updating scheme.

For example, let's say file *A* has been partitioned evenly into five parts (denoted by *a*, *b*, *c*, *d*, and *e*). Assume the accessing pattern of the file is *a, b, c, d, a, b, b, a, c, d, a, b, c, d, a, b, d, a, c, c, d, a, b, c*. The snapshot of storing each of the accessed data into the flash memory consisting of 10 blocks with 4 pages (each, sequentially) is shown in Figure 6. Firstly, the first four accessed data are stored sequentially in block b_1. When storing the second accessed data *d* (the 10th appearance data in the access pattern) into the second free page in block b_3, block b_1

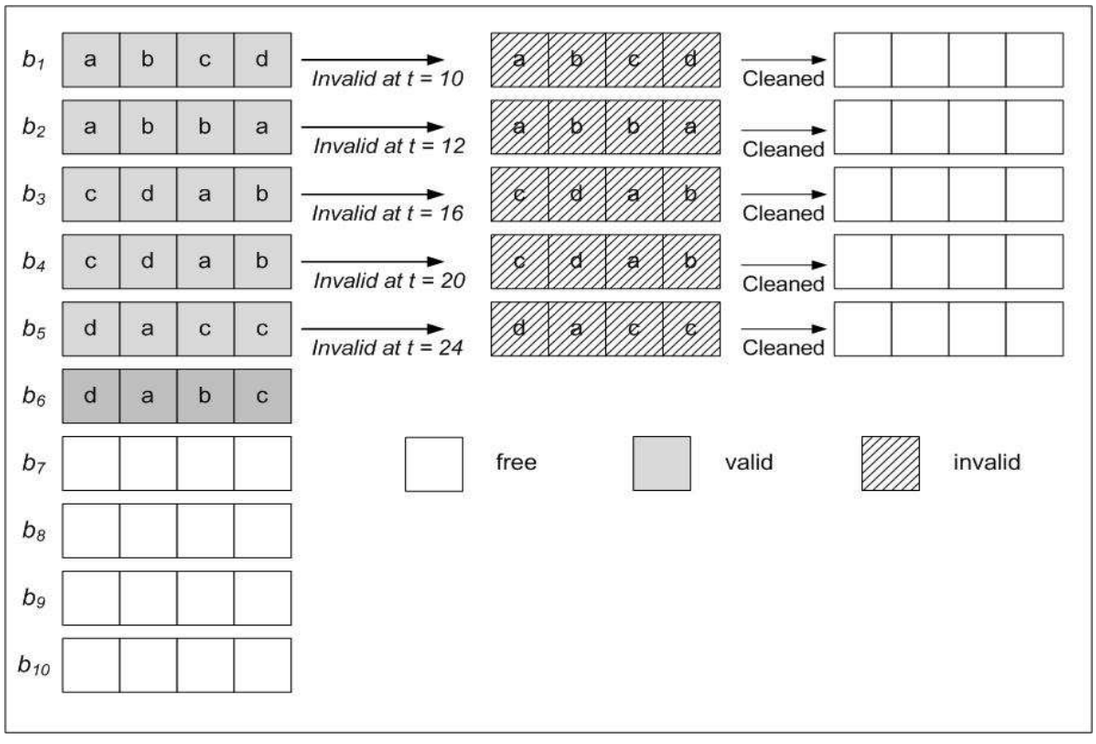


Fig. 6. Automatic cleaning process in sequential data allocation scheme.

turns into an inactive state and can be erased automatically. Then, block b_2 is erased when storing the last free page in block b_3 with data b. Block b_3 is erased when finish storing the 5th appearance of data b into the last free page of $b4$. At the end of the access pattern, only block b_6 is in the active state. When the inactive block is erased, all of its pages are changed into the free state and the block is ready for storing new or updated data.

4.2 Semi-automatic cleaning

Semi-automatic cleaning is commenced when the memory array free spaces reach a certain threshold, for instance, when the available free space is fewer than 20% – 35% of the total memory space. Two primary goals of the semi-automatic cleaning process are: 1) *Minimizing cleaning cost*, and 2) *Wearing blocks evenly*. Unlike the automatic cleaning process, single or multiple active block(s) can be cleaned simultaneously when the semi-automatic process has been initiated. Therefore, since the blocks to be cleaned contain valid data, the data needs to be migrated first before the cleaning process can be initiated and the current memory operations are temporarily halted. It is resumed when the process has ended. Besides, the cleaning cost required is inconsistent and it solely depends on the block utilization (u_i) level and the number of active blocks involved in the cleaning process. The cleaning cost is the total access time required to erase the victim blocks which includes several reads and writes accessing time (depending on the block utilization levels) plus the erasure time. In short, it can be simplified as in Equation 1 [17]. Block utilization is the ratio between valid pages and total pages.

$$T(b_i) = R_t + (W_t \times 10) + E_t \times 75 \tag{1}$$

In Equation 1, the write function is assumed to be 10 times slower than the read function while the erase function is 75 times slower than the read function. Figure 7 presents the cleaning cost required for cleaning a single victim block in the memory array. To illustrate this, assume a block containing 64 pages, and the block utilization level is between 0 and 100 %. The actual time for read, write and erase access functions were taken from Figure 3.

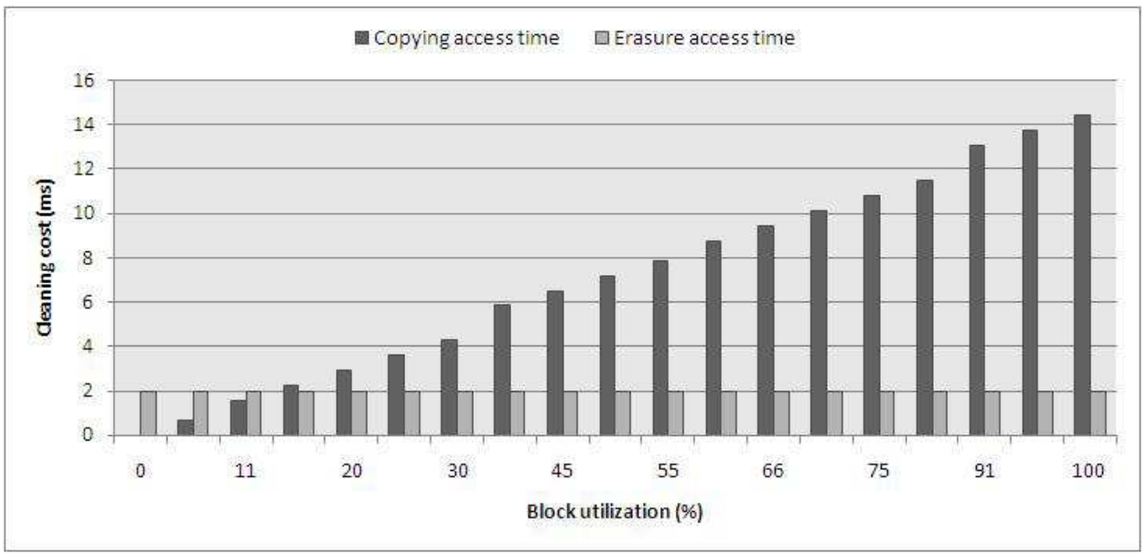


Fig. 7. The cleaning cost for single block.

As illustrated in Figure 8, the semi-automatic cleaning is undertaken in three stages. First, a victim block (b_1) to be cleaned is selected. Second, all valid pages residing in block b_1 are identified (e.g., *a, b, c,* and *d*) and copied/migrated into free pages in block b_3 (initially, b_3 is in an inactive state). In the last stage, block b_1 is erased when all the valid pages have been copied. Since multiple victim blocks can be erased simultaneously, the process could affect the current I/O operational functions. Therefore, the numbers of victim blocks becomes a crucial factor in the semi-automatic cleaning process. Unlike the automatic cleaning process, there are several important issues that need to be considered in semi-automatic cleaning. The four main issues in the semi-automatic cleaning process are 1) *Execution time*, 2) *Victim block selection procedure*, 3) *Victim block amount*, and 4) *Valid data re-organization* [18].

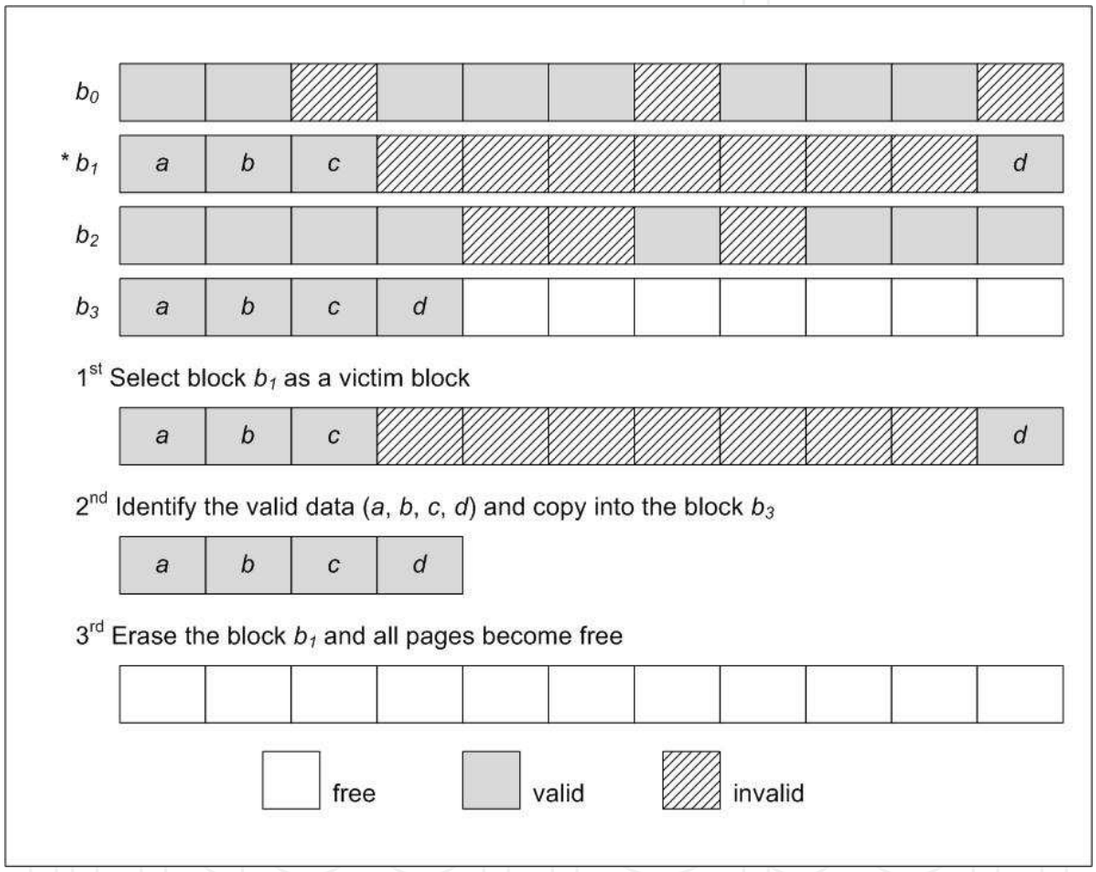


Fig. 8. Three stages in the semi-automatic cleaning process.

The execution time issue refers to the time to initiate the cleaning process, either periodically or according to memory free space availability. The victim block selection procedure refers to the method used to select the block to be erased and the straight forward approach is selecting a victim block that contains the largest amount of garbage. Other parameters include cost to erase, block lifespan, erasure count, and age of data [1, 10, 21, 22]. Again, the victim block amount issue in the semi-automatic cleaning enables single or multiple victim blocks to be erased simultaneously. On the other hand, both approaches have their own pros and cons. Cleaning a single block requires smaller access time but it also requires many erase operations. In contrast, erasing multiple blocks can distract the execution of normal

I/O operational system execution [18], but multiple victim blocks cleaning helps in reorganizing many valid data and can also help in reducing the number of blocks to be further erased. Then, the valid data re-organization issue refers to the process of copying the valid data in the victim block into a new free location in the available active blocks. The common approach is the valid data clustering technique, where valid data will be grouped into the similar block according to the data feature (such as regularly modified, irregularly modified, data time-stamp, and related data file). Thus, in order to improve the semi-automatic cleaning process performance, a number of studies that focuses on determining victim blocks have been proposed. The accompanying table shows the summary of the studies. In addition, the cleaning cost in the semi-automatic process depends on two important parameters, namely, 1) *Number of victim blocks* and 2) *Amount of valid data.* The cleaning cost will be extremely boosted when both parameters increase. However, the number of active blocks is not fixed and it is a controllable parameter. Due to this, by employing a proper allocation scheme, the amount could be minimized since the inactive block can be erased at the background.

Cleaning scheme	Victim block selection procedure/equation	Wear-leveling
Greedy (GR) [19]	$\cos t(B_i) = \frac{u_i}{1-u_i}$	No
Cost-benefit (CB) [20]	Block with maximum value from equation $a \times \frac{(1-u_i)}{2u_i}$	No
Cost age time (CAT) & Dynamic dAta clustering (DAC) [21]	Block with minimum value from equation $\left(\frac{u_i}{1-u_i}\right) \times \frac{1}{a} \times e$	Yes
Cost Age Time with Age Sort (CATA) [18]	Blocks those maximize equation $\left(\frac{1-u_i}{1+u_i}\right) \times a \times \frac{1}{e}$	Yes
S-Greedy (S-GR) [22]	Based on GR algorithm and focus on valid data distribution	Yes

u_i: block i utilization level. a: the last invalidation time in the block. e: block erasure count.

Table 1. A summary of previously proposed victim block selection algorithm.

5. Summary

Flash memory offers several superior features as a secondary storage and has recently been employed in many consumer electronic gadgets. However, due to the hardware operational characteristics, especially the out-place updating scheme, several challenges have emerged in terms of data management in designing and implementing an efficient data storage system. There are existing issues that influence flash memory performance, which are related to the cleaning process in order to allow data storage continuity. Both the automatic and the semi-automatic cleaning processes are two important issues in guaranteeing cleaning process performance in the flash memory. The automatic cleaning is directly

related with the efficient data allocation schemes where the cleaning can be initiated without having to disturb the current operations in the flash memory. Although only single inactive blocks can be cleaned every time the process is initiated, when the amount of active-to-inactive state conversion increases, the cleaning performance of the flash memory is guaranteed since the inactive block can be erased automatically without having to disturb current I/O operations. Conversely, the semi-automatic cleaning process is initiated according to a memory array free space threshold or it can be initiated periodically. There are several parameters employed in establishing the victim block to be erased such as cleaning cost, erasure count, age of data, block utilization, etc. Although the cleaning can be initiated on multiple victim blocks, the process can impose a blocking time that would distract the normal I/O operation execution on the memory. On the other hand, the efficiency of re-organizing the valid data in the victim blocks could influence the cleaning process performance further. The well-organized valid data in the new active block will group the regular and irregular accessed data into different blocks and could further increase the amount of inactive blocks. The increase of inactive blocks in the memory array would increase the automatic cleaning process and guarantee flash memory performance. Thus, both cleaning processes are important in order to improve the cleaning process performance in flash memory as well as its endurance.

6. References

[1] Douglis, F., Kaashoek, F., Marsh, B., Caceres, R., Li, K. and Tauber, J. (1994) Storage alternatives for mobile computers. In: Proceedings of the 1st USENIX Conference on Operating Systems Design and Implementation (OSDI'94), Nov. 14-17, Monterey, California: ACM/IEEE. pp. 25 – 37.

[2] Chang, L.P. and Kuo, T.W. (2004) An efficient management scheme for large-scale flash memory storage systems. In: Proceedings of the 2004 ACM Symposium of Applied Computing (SAC'04), March 14-17, Nicosia, Cyprus: ACM. pp. 862 – 868.

[3] Lawton, G. (2006) Improved flash memory grows in popularity. IEEE Computer, 39(1), p. 16 – 18.

[4] Lim, S.H. and Park, K.H. (2006) An efficient NAND flash file system for flash memory storage. IEEE Transactions on Computers, 55(7), p. 906 – 912.

[5] Breeuwsma, M., Jongh, M.d., Klaver, C., Knijff, R.v.d. and Roeloffs, M. (2007) Forensic data recovery from flash memory. Small Scale Digital Device Forensic Journal, 1(1), p. 1 – 17.

[6] Hsieh, J.W., Tsai, Y.L., Kuo, T.W. and Lee, T.L. (2008) Configurable flash-memory management: Performance versus overheads. IEEE Transactions on Computer, 57(11), p. 1571 – 1583.

[7] Woodhouse, D. (2001) JFFS: The journaling flash file system. In: Proceedings of the 2001 Ottawa Linux Symposium, July 13-16, Ottawa, Canada.

[8] Barre, A.G. (1993) Flash memory magnetic disk replacement? IEEE Transactions on Magnetics, 29(6), p. 4104 – 4107.

[9] Sharma, A.K. (2003) Advanced semiconductor memories: Architecture, designs, and applications. Canada: WILEY-IEEE Press. P.4

[10] Kawaguchi, A., Nishioka, S. and Motada, H. (1995) Flash memory based file system. In: Proceedings of USENIX 95 Technical Conference, Jan. 16-20, New Orleans, Louisiana: USENIX. pp. 155 – 164.

[11] Wu, M. and Zwanepoel, W. (1994) eNVy: a non-volatile, main memory storage system. In: Proceedings of the 6th International Conference on Architectural Support for Programming language and Operating Systems (ASPLOS), Oct. 5-7, San Jose, California: ACM. pp. 86 – 97.

[12] Chou, L.F. and Liu, P. (2005) Efficient allocation algorithms for flash file systems. In: Proceedings of 11th International Conference on Parallel and Distribution Systems (ICPADS'05), July 20-22, Fukuoka, Japan: IEEE. pp. 634 – 641.

[13] Liu, P., Chuang, C.H. and Wu, J.J. (2007) Block-based allocation algorithms for flash memory in embedded systems. In: Proceedings of 9th International Conference on Parallel Computing Technologies (PaCT 2007), Sept. 3-7, Pereslavl-Zalessky, Russia: Springer. pp. 569 – 578.

[14] Kim, H. and Lee, S.G. (2002) An effective flash memory manager for reliable flash memory space management. IEICE Trans. Information and System, E85-D(6), p. 950 – 964.

[15] Chang, Y.H., Hsieh, J.W. and Kuo, T.W. (2007) Endurance enhancement of flash-memory storage systems: An efficient static wear leveling design. In: Proceedings of 44th ACM/IEEE Design Automation Conference (DAC 2007), June 4-8, San Diego, California: ACM. pp. 212 – 217.

[16] Rahiman, A.R. and Sumari, P. (2009). Probability based page data allocation scheme in flash memory. In: Proceedings of IEEE Pacific-Rim Conference on Multimedia (PCM 2009), Dec. 15-18, Bangkok, Thailand: IEEE. pp. 300 – 310.

[17] Ko, S., Jun, S., Kim, K., and Ryu, Y. (2008) Study on garbage collection schemes for flash based Linux swap system. In: International Conference on Advanced Software Engineering & Its Applications (ASEA 2008), Dec. 13-15, Hainan Island, China: IEEE. pp. 13 – 16.

[18] Han, L.Z., Rhu, Y., Chung, T.S., Lee, M. and Hong, S. (2006) An intelligent garbage collection algorithm for flash memory storages. In: Proceedings of International Conference on Computational Science and Its Applications (ICCSA 2006), May 8-11, Glasgow, UK: Springer. pp. 1019 – 1027.

[19] Rosenblum, M. and Ousterhout, J.K. (1992) The design and implementation of a log-structured file system. ACM Transactions on Computer Systems, 10(1), p. 26 – 52.

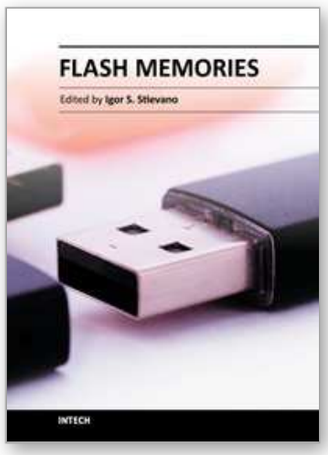
[20] Kawaguchi, A., Nishioka, S. and Motada, H. (1995) Flash memory based file system. In: Proceedings of USENIX 95 Technical Conference, Jan. 16-20, New Orleans, Louisiana: USENIX. pp. 155 – 164.

[21] Chiang, M.L., Lee, P.C.H, and Chang, R.C. (1999) Cleaning policies in mobile computers using flash memory. Journal of Systems and Software, 48(3), p. 213 – 231.

[22] Kwon, O., Ryu, Y. and Koh, K. (2007) An efficient garbage collection policy for flash memory based swap systems. In: Proceedings of International Conference on Computer Science and Applications (ICCSA 2007), Oct. 24-26, San Francisco, USA: IAENG. pp. 213 – 223.

[23] Yaffs (2006) How does YAFFS work? [Online], [Accessed 30th July, 2010], Available from World Wide Web: http://www.yaffs.net/yaffs-internals.

[24] Kang, J.U., Kim, J.S., Park, C., Park, H. and Lee, J. (2007) A multi-channel architecture for high-performance NAND flash-based storage system. Journal of Systems Architecture, 53(9), p. 644 – 658.



Flash Memories

Edited by Prof. Igor Stievano

ISBN 978-953-307-272-2
Hard cover, 262 pages
Publisher InTech
Published online 06, September, 2011
Published in print edition September, 2011

Flash memories and memory systems are key resources for the development of electronic products implementing converging technologies or exploiting solid-state memory disks. This book illustrates state-of-the-art technologies and research studies on Flash memories. Topics in modeling, design, programming, and materials for memories are covered along with real application examples.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Amir Rizaan Rahiman and Putra Sumari (2011). Block Cleaning Process in Flash Memory, Flash Memories, Prof. Igor Stievano (Ed.), ISBN: 978-953-307-272-2, InTech, Available from: http://www.intechopen.com/books/flash-memories/block-cleaning-process-in-flash-memory

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen