A Long Endurance Policy (LEP): An Improved Swap Aware Garbage Collection For NAND Flash Memory Used As A Swap Space In Electronic Devices

Arushi Agarwal , Surabhi Maddhesiya, Priya Singh and Rajendra Kumar Dwivedi

Abstract --- Flash memory has more capacity and less weight. It makes it more suitable for electronic media. Electronics such as tablet PC and smart phones use NAND flash memory as a secondary storage because it has many attractive features such as small size, fast access speeds, and light weight. However, it has shown limited success in its battle against the hard disk, due to intrinsic weak points of: erase-before-rewrite and limited program/erase cycles. Electronics with NAND flash memory uses a "swapping mechanism" to extend a limited main memory space. However, if the electronic devices use flash memory as swap space, it should perform garbage collection, which is a time consuming operation. Therefore, in order to manage swap space efficiently along with minimizing the weak points, this work presents a novel policy that improves the lifetime and provides efficient garbage collection for those devices. The proposed policy has three features important in NAND flash memory based swap systems: (1) long endurance of flash memory, (2) quick garbage collection time, and (3) low energy consumption.

Index terms – Electronic devices, flash translation layer, garbage collection, NAND flash memory, Swapping, swap space, wear-leveling.

1. INTRODUCTION

In recent years flash memory has more capacity and a lower price. It makes flash memory more suitable for portable consumer electronics. For example, most digital music players work with NAND flash memory. Now, makers of smart phones and portable game players use the flash memory technology or plan to exploit it in the near future because it has many attractive features such as small size, shock resistance, high reliability, low power consumption, and light weight [16].

Typical electronics such as cellular phones and digital music players contain DRAM, NOR flash and NAND flash memory. DRAM is used for a main memory, NOR flash memory is used for a program code, and NAND flash memory is used for user data. Because the devices contain three kinds of memory, it is difficult to cut down the cost of hardware and reduce the size of it. In order to reduce the cost and size, it has been attempted to eliminate NOR flash memory from the electronic devices. Since the device does not contain NOR flash memory, the application program code needs to be copied from NAND flash memory to the main memory during running the application. So, "demand paging" is exploited.

Demand paging is a virtual memory technique that code or data is loaded from the secondary storage only when it is needed by a process. Furthermore, the portable consumer electronics using demand paging can exploit a "swapping" mechanism to extend a limited main memory space because the mobile game codes become large in recent years. Since, the swapping frequently performs read and write operations to the swap space, there are many invalid pages to free pages. In NAND flash memory, especially, the number of erase operations is limited to about 100,000 times. Because of this reason, an efficient policy should be developed for NAND flash memory based

Fig.1 Electronic devices such as smart phones, portable game players, and tablet PC. The makers of these consumer electronics now use flash memory.

I. Arushi Agarwal Email- talk2arushi@gmail.com
II. Surabhi Maddhesiya , Email- surabhi.2907@gmail.com
III. *Priya* Singh , Email- sweetpriya.singh32@gmail.com
IV. Rajendra Kr. Dwivedi , Email- rajendra_bhilai@yahoo.com
M.M.M.Engg.College, Gorakhpur,India.

swap system. This paper proposes a new policy for the electronic media with swapping mechanism. The proposed policy focuses on minimizing the garbage collection time, reducing the energy consumption, and extending the lifetime of NAND flash memory to improve a performance of electronic media. Trace-driven simulations show that the proposed policy performs better than the existing garbage collection such as the Greedy, the Cost-Benefit (CB), and the Cost Age Time (CAT) policies in

TABLE 1
PERFORMANCE COMPARISON BETWEEN NOR AND NAND FLASH MEMORY [15]

	NOR multilevel cell (Mbytes/sec)	NAND 90-nm single-level cell(x8, large block) (Mbytes/sec)	Samsung OneNAND 90-nm(Mbytes/sec)
Read	108	16.2	108
Write	0.14	6.8	8.2
Erase(single)	0.11	64	64
Erase(multiple)	0.11	NA	2

terms of the endurance of flash memory, the garbage collection time, the number of erase operations, and the energy consumption.

2. LITERATURE SURVEY

This section describes some important characteristics of flash memory specially focusing on the viewpoint of garbage collection and analyzes existing garbage collection policies.

2.1 Characteristics of NAND Flash Memory

Flash memory is a non-volatile solid state storage whose density and I/O performance have improved to a level at which it can be used as an auxiliary storage for mobile embedded devices. There are two types of popular flash memory, namely NOR and NAND flash memories. NOR flash memory allows for efficient random of byte data similar to DRAM. Hence, it is useful for execution-in-place (XIP) of program codes. In contrast, NAND flash memory is more suitable for storing bulk data such as multimedia files. The reason is that NAND flash memory offers high density and fast I/O operations for a group of data called flash page. As a result, NAND flash memory is preferably used as a storage system of portable media players and mobile computers. Recently, hybrid flash memory products that contain RAM components in NAND flash memory are emerging and the domain of NAND flash memory is growing as well. For example, Samsung's OneNAND combines a NAND core with SRAM. It could execute program codes like NOR flash memory and offers higher data density like NAND flash memory [16]. The performances of NOR, NAND, and hybrid flash memory are summarized in Table 1.

Flash memory is partitioned into blocks and each block has a fixed number of pages. Unlike hard disks, flash memory has three kinds of operations: page read, page write, and block erase. As shown in table 1, each operation has significantly different performance characteristics. With these characteristics, flash memory has two drawbacks related to I/O operations. First, block of flash memory need to be erased before they are rewritten. An erase operation needs more time and energy than a read or a write operation.

The second drawback is that the number of erase operations allowed to each block is limited. This drawback becomes an obstacle to develop a reliable flash memory based embedded systems. To relieve this problem, the software layer of a flash memory device usually contains wear-leveling mechanism that controls the erase count of all blocks as evenly as possible.

2.2 Existing Works on Garbage Collection

Rosenblum et al. proposed the Log-Structured File System (LFS) which supports out-place-update I/O operations for disk storage to improve the I/O performance. Due to out-place-update property of LFS, garbage collection policies have already been discussed in disk storage systems [1-4]. Fortunately, the LFS and log based disk storage can be applied to NAND flash memory based storage systems. Wu et al. proposed the Greedy policy for garbage collection. The Greedy policy considers only valid data pages in blocks to reduce write cost and chooses the block with least utilization. However it does not consider wear leveling.

Kawaguchi et al. proposed the Cost Benefit policy [6]. This policy evaluates the cost benefit of all blocks in flash memory using ((a*(1-u))/2u) method, where a is the elapsed time from the last data invalidation on the block, and u is the percentage of fullness of the block. After evaluating all blocks, it chooses the victim block that has maximum cost benefit value.

Chiang et al. proposed the Cost Age Time (CAT) policy [7]. This policy focuses on reducing the number of erase operations. To reduce the number of erase operations, they use a data redistribution method. The method operates at the granularity of pages. Furthermore, the CAT policy considers wear-leveling.

Kim et al. proposed the cleaning cost policy [9], which focuses on lowering cleaning cost and evenly utilizing flash memory blocks.

2.3 Garbage Collection Procedure

In order to improve I/O performances, flash memory based systems support out-place-updates, which needs garbage collection. The garbage collection procedure is performed by the following steps:

1. Select one out of the used blocks as a victim block.

2. If there are valid pages in the victim block, they are copied out of the free block.
3. Erase the victim block after copying out the valid pages.
4. Add the erase block to the free block list.

To make sufficient free spaces in the flash memory, garbage collection operation is performed repeatedly, which needs many write and erase operations. In order to improve the I/O performance during this procedure, effective victim selection is important. Fig. 2 illustrates an

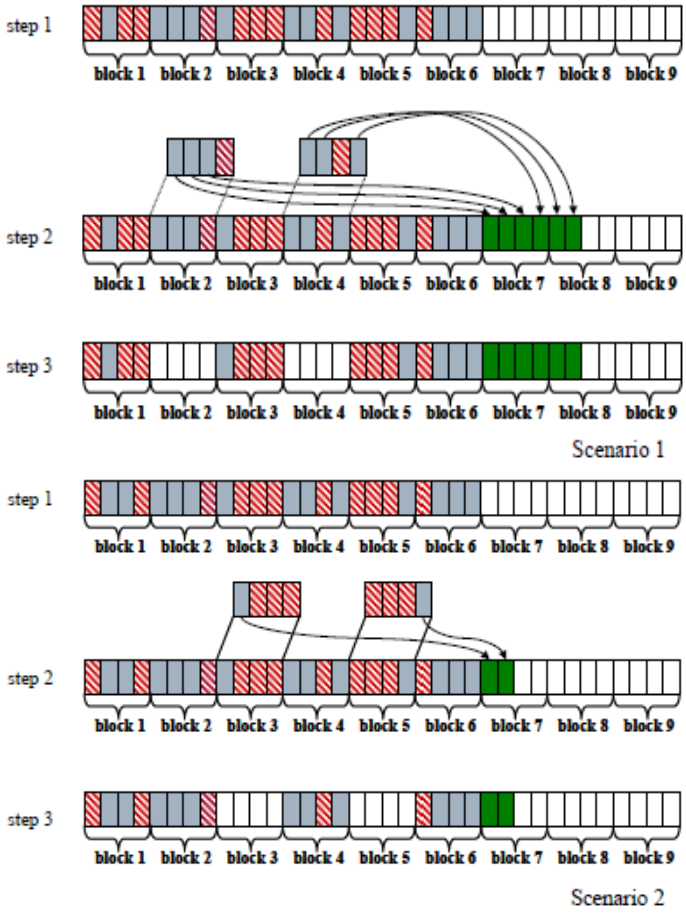

Fig. 2 Examples of garbage collection procedure. These examples show the reason why the victim selection is important.

example of garbage collection procedure, and shows why victim selection is important. There are six used blocks and three free blocks in Fig. 2. The garbage collection procedure will be performed to make more free blocks in the example. In the scenario 1, the garbage collector selects block 2 as victim block, copies out the valid pages to block 7 that is a free block, and then erases block 2 to make a free block. After erasing block 2, garbage collection is performed again. At this time, the garbage collector selects block 4, copies out valid pages, and erase block 4. Although the garbage collector performed six write operations and two erase operations, there are still six used blocks (block 1, 3, 5, 6, 7, and 8) and three free blocks (block 2, 4, and 9). In other words, the procedure did not make another free block. Therefore, the garbage collector should perform garbage collection repeatedly until free blocks are sufficient in the flash memory. In contrast with scenario 1, the garbage collector of scenario 2 selects block 3 that owns only one valid page as a victim block and copies out the valid page to block 7. Then it erases block 3 to make a free block. After erasing block 3, it selects block 5, copies out the valid pages, and erase block3. Hence, the garbage collector performed two write operations and two erase operations in scenario 2. Though scenario 2 performed less write and erase operations than scenario 1, it made five used blocks (block 1, 2, 4, 6, and 9) and four free blocks (block 3, 4, 8, and 9). This is because scenario 2 selected a block with the least number of valid pages as the victim block.

3. Swap Space Management Technique

Portable consumer electronics using demand paging exploit a "swapping" mechanism to extend a limited main memory space. The swapping mechanism frequently performs read, write, and erase operations to the swap space. Specifically, in flash memory, write and erase operations are even slower and also needs more energy than a read operation. Thus, the write and erase operations are dominant to I/O performances of flash memory based swap systems. Ohhoon, K et al. proposed a garbage collection policy named as "Swap Aware Garbage Collection" to manage the swap space efficiently and improves lifetime of NAND flash memory.

3.1 NAND flash memory based swap system

The architecture of NAND flash memory based swap system consists of four parts. First, the 'swap space' consists of a sequence of page slots, which is used to store a page swapped out from the main memory. When a page is swapped out, its location is stored in the corresponding 'page table entry (PTE)' [Second]. The information in the PTE is used to find the correct swap slot in the swap space when the page is swapped in. Unlike a hard disk based swap system, the NAND flash memory based swap system has the Flash Translation Layer (FTL) and the Memory Technology Device (MTD) layer. Third, the 'FTL' provides a transparent access to flash memory system. If there are not enough spaces in the swap space, the swap system should perform garbage collection. Garbage collection is also handled in FTL [13]. Fourth, the 'MTD' layer handles read, write, and erase operations for the swap system [14].

3.2 Block recycling for swap space

While garbage collection is performed, other I/O operations are blocked. This may incur degradation of I/O performances. Hence, minimizing the garbage collection time is important to improve the I/O performance of NAND flash memory. It basically tried to select the victim block to be erased similar to the Greedy policy during garbage collection. Because the Greedy policy selects the block that requires the least number of write operations, it could minimize the garbage collection time. However,

since the Greedy policy does not consider wear-leveling, it shows poor performance in terms of the endurance of NAND flash memory for references with high spatial locality. To address the problems, it considered the invalidation time of invalid pages, the swapped out time of valid pages, and the erase count of flash blocks.

When garbage collection is needed, the policy calculated the *Invalid Age (IA)* of each block, and selects the block with the largest IA as the victim block. The IA of a block is computed as-

$$IA = \sum_{i=1}^{n} i\_age_i$$

$$i\_age_i = c\_time - i\_time_i \qquad (1)$$

where n is number of invalid pages in a block, *c_time* is current time, and i_time_i is time when status of page i is changed to "invalid". Hence, i_age_i is the elapsed time since page i becomes invalid. If there are many invalid pages in a block, the IA of the block is large because i_age's of all invalid pages are added. Since the garbage collector selects the block with the largest IA as the victim block, this reduces the copy-out cost of valid pages. On the other hand, even if a certain block has only one invalid page, the i_age of the page could be large enough if the page was invalidated long time ago. In this case, the IA of the block becomes large and it can be selected as victim. This eventually improves wear-leveling of the flash memory.

After selecting victim blocks, the next step was to copy valid pages of the victim blocks to free blocks. In this procedure, "redistribution of valid pages" was performed.

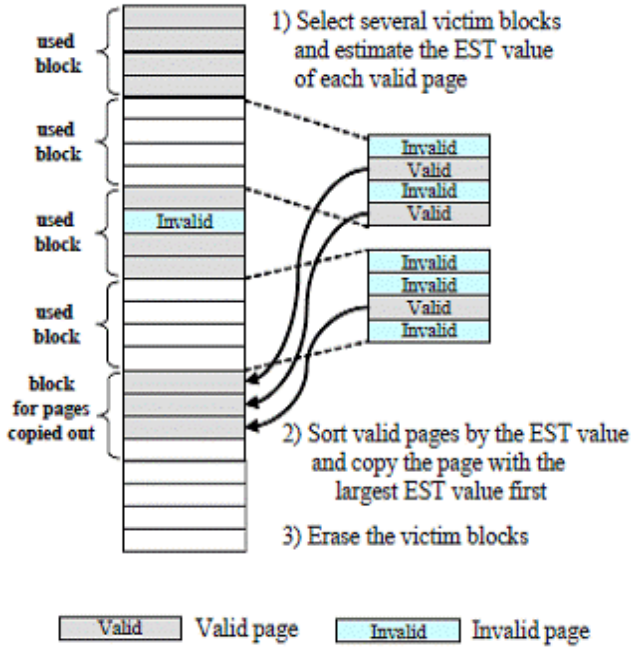

Fig. 3 The redistribution of valid pages.

For the redistribution of the valid pages, the policy considered the *Elapsed Swapped-out Time (EST)* of the valid page. The *EST* of each valid page is computed as

$$EST_i = c\_time - s\_time_i \qquad (2)$$

Where *c_time* is current time, and s_time_i is time when $page_i$ is swapped out from the main memory. Hence, EST_i is the elapsed time since page i is swapped out from the main memory. Because the current operating system use the round-robin based process scheduling scheme, the least recently swapped-out page is likely to swap in the main memory in the near future [11].

Thus, the recently swapped-out page is classified as "hot page" in the policy. The policy could get the hot valid pages together into a block during redistribution because the policy sorted the valid pages by the EST value, and then copied out the least recently swapped out page first. Fig. 3 shows the redistribution of valid pages during garbage collection.

3.3 Free block management for swap space

As already mentioned, the number of erase operations allowed to each block is limited. Thus, the flash memory should be controlled to evenly wear out all blocks since wearing out specific blocks could limit the usefulness of the whole flash memory. In order to address this problem, most of existing garbage collection policies consider wear-leveling of flash memory when the victim block is selected. In contrast, that policy presented an efficient free block management scheme for wear leveling of NAND flash memory to guarantee long endurance of swap system. Fig. 4 shows the free block management scheme for wear leveling. The policy exploited the sorted free block victim blocks, it calculated the number of the erase operation of the erased block, and then the block is added to the free block list. The free block in the free block list were sorted by the number of erase operations of the block. During garbage collection, the block with the minimum number of erase counts is allocated to valid pages. This newly allocated block is called "active block" in figure 4. This mechanism actually controlled wear-leveling of the NAND flash memory based swap system. For efficient implementation, the min heap data structure can be used to find the block with minimum erase count.

4. A PROPOSED "LONG ENDURANCE POLICY"

After calculating the IA of each block, the previous proposed policy, SAGC selects several victim blocks with the largest IA and then copies valid pages in the victim blocks to free blocks. SAGC policy selected a random number of victim blocks which may result in only few number of free blocks added to the free block list at the end of garbage collection procedure. **And, as a consequence, garbage collection is needed again and again which may be a wastage of time as in each call, IA value of each block is calculated, in order to select victim blocks and**

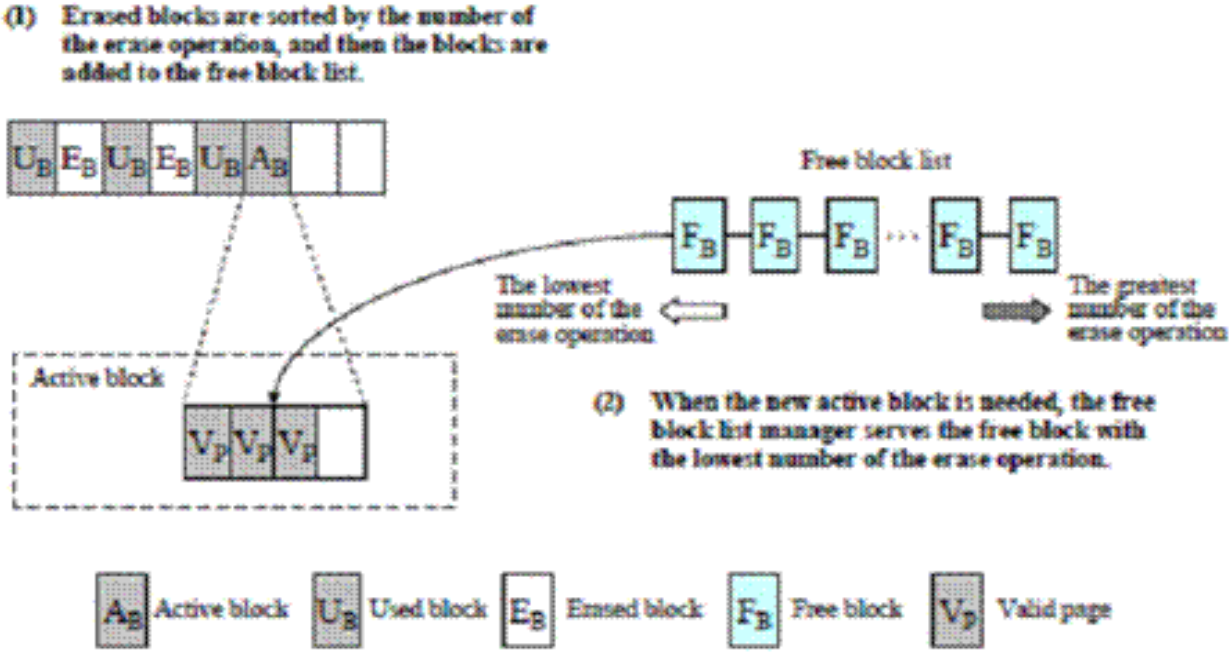

Fig. 4 The efficient free block list management for wear-leveling.

also function overheads associated with each call. But, the number of victim blocks selected after calculating IA, also affects the performance of the system.

So, the proposed "Long Endurance Policy", considers that the number of victim blocks selected at a time (n_victim) depends on the number of free block at that time (n_free). Let, minimum number of free blocks needed when garbage collection stops is n_min then

If n_free < (n_min – n_free)
Then n_victim = n_free
Else n_victim = 2*(n_min- n_free)

For example, consider that to evaluate the performance, when the size of free block is fewer than 10% of the total size of flash memory, garbage collection is started. And the garbage collection is stopped when the size of free block list is larger than 20% of the total size of the flash memory. Suppose a flash memory consists of 100 blocks out of which just 9 blocks are free at an instant, which is less than 10% so garbage collection is needed. According to this proposed policy, initially the number of victim blocks selected is 9 as 9 < (20 – 9). Supposing that after copying valid pages and erasing the victim blocks, the number of extra free blocks added is 6. But till now size of free block list has not reached to 20% of the size of flash memory so once again garbage collection is needed. This time number of victim blocks selected is 10 as 15 > (20 – 15). Supposing that after copying valid pages and erasing the victim blocks, the number of extra free blocks added is 6. Now size of free block list has reached 21 and hence, no need of garbage collection procedure for the instant.

Thus, it is seen that number of times garbage collection procedure is called is just 2 whereas random number of victim blocks selection may lead to a lot many times calling of the garbage collection procedure. Though number of write and erase operations do not change but then also the time taken by various other operations (like computation of IA each time) will be reduced in this proposed policy. Hence, efficient number of victim blocks selection reduces the number of times garbage collection is called.

This proposed policy eventually improves the endurance of NAND flash memory based swap system. Thus, the proposed garbage collection policy is named as "Long Endurance Policy (LEP)".

The operations for the NAND flash memory based system are described in Fig. 5

```
/* If there are not enough free spaces in the main memory,
   the swappable page is removed from the main memory
   and then, it is copied into the swap area.
*/
Swap_out()
{
     Allocate a free block for the swapped-out page;
    Write the page into the free block;
}

/* If the process accesses the swapped-out page,
   The swapped-out page is copied into the main memory.
*/


Swap-in()
{
     Copy the swapped-out page into the main memory;
     Mark the obsolete page as invalid in the swap area;
```

```
}

/* If there are not enough free spaces in the swap area the
   garbage collection is performed to translate the invalid
   space to the free space
*/
Garbage_Collection()
{
    Calculate the IA of all blocks;
    Select proper number of victim blocks with largest IA;
    For all the valid data pages in the victim blocks
    {
       Calculate the EST value of valid pages;
       Sort the valid pages by the EST value;
       Copy the page with largest EST value into the
       active block first;
    }
    Erase the victim blocks;
    Add the erased blocks to the free block list;
    Sort the blocks in the free block list by the number of
    erase operations;
}
```

Fig. 5. Operations for NAND flash memory based swap system.

5. CONCLUSION

This paper presents a novel swap space management scheme, which is specifically designed for NAND flash memory based electronics devices such as digital music players, portable game players, and smart phones. To manage the swap space efficiently, this paper proposed the new garbage collection policy for the electronic devices. The proposed policy has three features important in NAND flash memory based swap systems: (1) long endurance of flash memory, (2) quick garbage collection time, and (3) low energy consumption.

In order to minimize the garbage collection time, reducing the energy consumption, and extend the lifetime of the NAND flash memory, the proposed policy considers the invalidation time of invalid pages, the swapped-out time of valid pages, and the erase count of flash blocks. As a result, the proposed policy performs better than other existing policies in terms of number of erase operations, the garbage collection time, total energy consumption and the endurance of NAND flash memory.

6. REFERENCES

[1] Rosenblum, M., Ousterhout, J.K., "The Design and Implementation of a Log-Structured File System," ACM Transactions on Computer Systems, Vol. 10, No. 1, 1992.

[2] Blackwell, T., Harris, J., Seltzer, M., "Heuristic Cleaning Algorithms in Log-Structured File Systems," Proceedings of the 1995 USENIX Technical Conference, Jan. 1995.

[3] Matthews, J. N., Roselli, D., Costello, A. M., Wang, R. Y., Anderson, T. E., "Improving the Performance of Log-Structured File Systems with Adaptive Methods," Proceedings of the Sixteenth ACM Symposium on Operating System Principles, 1997.

[4] Seltzer, M., Bostic, K., McKusick, M. K., Staelin, C., "An Implementation of a Log-Structured File System for UNIX," Proceedings of the 1993 Winter USENIX, 1993.

[5] Wu, M., Zwaenepoel, W., "eNVy: A Non-Volatile, Main Memory Storage System," Proceedings of the 6th International Conference on Architectural Support for Programming Languages and Operating Systems, 1994.

[6] Kawaguchi, A., Nishioka, S., and Motoda, H., "A Flash-Memory Based File System," Proceedings of USENIX Technical Conference, 1995.

[7] Mei-Ling Chiang, Paul C. H. Lee, Ruie-Chuan Chang, "Cleaning policies in mobile computers using flash memory," Journal of Systems and Software, Vol. 48, 1999.

[8] Torelli, P., "The Microsoft Flash File System," Dr. Dobb's Journal, Feb. 1995.

[9] Hanjoon Kim, Sanggoo Lee, S. G., "A new flash memory management for flash storage system," Proceedings of the Computer Software and Application Conference, 1999.

[10] Li-Pin Chang, Tei-Wei Kuo, Shi-Wu Lo, "Real time garbage collection for flash-memory storage systems of real-time portable consumer electronics," ACM Transactions on Embedded Computing Systems, Vol. 3. 2004.

[11] D. P. Bovet and M. Cesati, "Understanding the Linux Kernel" O'Reilly, third edition, 2006.

[12] Samsung Electronics: 128M x 8 Bit NAND Flash Memory. http://www.samsung.com

[13] Intel Corporation, "Understanding the Flash Translation Layer (FTL) Specification".

[14] Yaghmour, K., "Building Embedded Linux Systems," O'Reilly & Associates, Inc. 2003.

[15] Santarini, Michael, "NAND versus NOR. Which flash is best for booting your next system," Electronics Design, Strategy, News (EDN), 2005.

[16] Lawton, G. Improved flash memory grows in popularity. IEEE Computer. 2006.

[17] Ohhoon, K., Kern, K., "Swap Space Management technique for Portable Consumer Electronics with NAND flash memory", IEEE Transactions on Consumer Electronics, Vol. 56, No. 3, 2010.