Journal of Engineering and Applied Sciences 13 (4): 1023-1028, 2018
ISSN: 1816-949X
© Medwell Journals, 2018

Sampling-Based Block Erase Table Method in Wear Leveling Technique for Flash Memory

Myungsub Lee
Department of Computer Engineering, College of Science and Technology,
Yeungnam University, Daegu, Korea

Abstract: We propose a novel wear-leveling technique using Sampling-based Block Erase Table (SBET). SBET relates one bit of the bit array table to each block by using exclusive OR operation with a round robin function. Accordingly, SBET enhances the accuracy of the cold block information and can prevent the decrease in performance of wear leveling. In the experiment, SBET prolonged the lifetime of flash memory by up to 88% compared with previous techniques which use a bit array table.

Key words: Flash memory, wear leveling, storage device, block erase table, round robin, robin function

INTRODUCTION

Flash memory is composed of multiple blocks and blocks are composed of pages. A page is composed of an array of cells that can store bits. The unit of writing/reading based on NAND is a page and the erasing unit is a block. In addition to that fact that flash memory involves different units for writing/reading and erasing it has the disadvantage of having to erase before writing this causes issues when refreshing data. In order to update one bit of data, all data in the block must be deleted and rewritten with the renewed data because the erasing unit is a block. This is an inefficient method compared to that of hard disks. This is called a “non-in place problem” of flash memory (Lawton, 2006). To resolve the non-in-place problem, the updated data is recorded in a different address and this address is then stored in a table and used for reference. This method of using address tables is divided into page mapping, block mapping and hybrid mapping according to the address unit. In page mapping, the change information of each page is composed and recorded as a combination of a logical address and a physical address. In block mapping, only block addresses are used and hybrid mapping involves the use of both the page address and the block address. The software layer that attempts to resolve the shortcomings of the flash memory using address mapping techniques and enables the use of hard disk-based application programs is called the Flash Translation Layer (FTL). In addition to address mapping tables, FTL includes a function for recovery after sudden system malfunctions, a garbage collection function which collects invalid pages and converts them into free pages and wear leveling, among other functions (Kwon *et al*., 2011; Chung *et al*., 2009; Ma *et al*., 2014).

To explain the wear leveling technique, we will first describe the wearing of flash memory. Flash memory applies a strong voltage to write and erase bits onto and from the cells this process damages the insulator inside the cells. Therefore, the per-block P/E cycle (Program/Erase cycle) is limited in order to prevent bit errors due to the damage of the insulator. NAND is divided into Single Level Ccell (SLC), MultiL Level Cell (MLC) and Triple Level Cell (TLC) according to the number of bits that can be stored in a cell and each type cell can store 1-bits, respectively. However, since, more accuracy is required from the hardware when more bits are stored, the available P/E cycle decreases. Typically, SLC, MLC and TLC each have a P/E limits of 10^5, 10^4 and 10^3 times, respectively. Therefore, since, the available P/E cycle decreases as the NAND flash memory becomes larger in capacity it is important to maintain the P/E at a uniform level (Lee and Kim, 2014; Park *et al*., 2012; Yang *et al*., 2014).

Wear leveling is a technique that extends the overall service life by evenly erasing each block of flash memory. Recently, various wear leveling methods have been researched to extend the life of flash memory. In most studies, wear leveling is performed by storing each block’s erasing count, time elapsed for referencing and erasing and reference frequency of each page, etc., into a memory table. However, it is difficult to apply embedded systems or sensor nodes with insufficient memory, because considerable memory is used by the tables for storing various data. Thus, a Block Erase Table (BET) technique was proposed in which the erasing of each block is stored in a table with a bit array and the memory usage is minimized to use it for wear leveling (Xu *et al*., 2014). BET reduces memory usage by using the value of “T” to control the frequency of wear leveling and uses the value of "k" to have multiple blocks correspond to a single bit. However, as memory usage decreases according to the increase in the value of "k" of BET, wear leveling performance rapidly decreases. In overcoming BET's shortcomings, the cause of this performance drop was attributed to a hidden cold block problem and a technique that uses the Bit Set Threshold (BST) calculated based on the number of invalid pages has been proposed (Kim *et al*., 2015). However, BST is limited when it comes to distinguishing cold blocks from invalid page count and has a disadvantage of high computation cost. This study proposes Sampling-based BET (SBET) using eXclusive OR (XOR) operation and a round-robin method to solve this problem. Considering how distinguishing cold blocks becomes more difficult as multiple blocks correspond to a single bit when the value of "k" increases, SBET uses XOR operation and the round-robin method to have each bit of BET correspond to a single block.

MATERIALS AND METHODS

SBET wear leveling technique

Sampling-based BET: Figure 1 shows an example of using an XOR-based round-robin index to have one block correspond to each bit in the BET. As shown in Fig. 1, each group is composed of four blocks and four BET bits. During the first wear leveling period, the round-robin index was set to 0, therefore, the blocks related to the XOR-based round-robin index that was acquired through the XOR index of each group corresponds to the BET bits. The remaining three irrelevant internal addresses were not included in the BET bit settings. When the second wear leveling occurs, the round-robin index will become 1 and the XOR-based round-robin index will be calculated again, so that, the blocks in each group correspond to the BET bits. When this method is used, the round-robin index returns to 0 and each block in the group repeatedly correspond to the BET bits in order. Therefore, if the XOR-based round-robin index is used, even if the irrelevant blocks in the group are erased, the cold blocks can be distinguished using the BET bits after a certain period because the BET bits were never established in the first place.

Wear leveling operation: As two or more blocks correspond to a single bit when the value of "k" is 1 or higher in the BET, SBET solves the hidden cold block problem by having one bit correspond to one block using the address in the group and the XOR-based round-robin index. The SWL-procedure-SBET algorithm 1 and 2 shows the wear-leveling process. The total number of block erases (e_{cnt}) in line 2 is divided by the number of 1 bits (f_{cnt}) in the BET and if this value is larger than "T," then the wear-leveling process begins. Lines 11-12 searches for 1 bit in the BET and delivers the relevant bit number (f_{index})

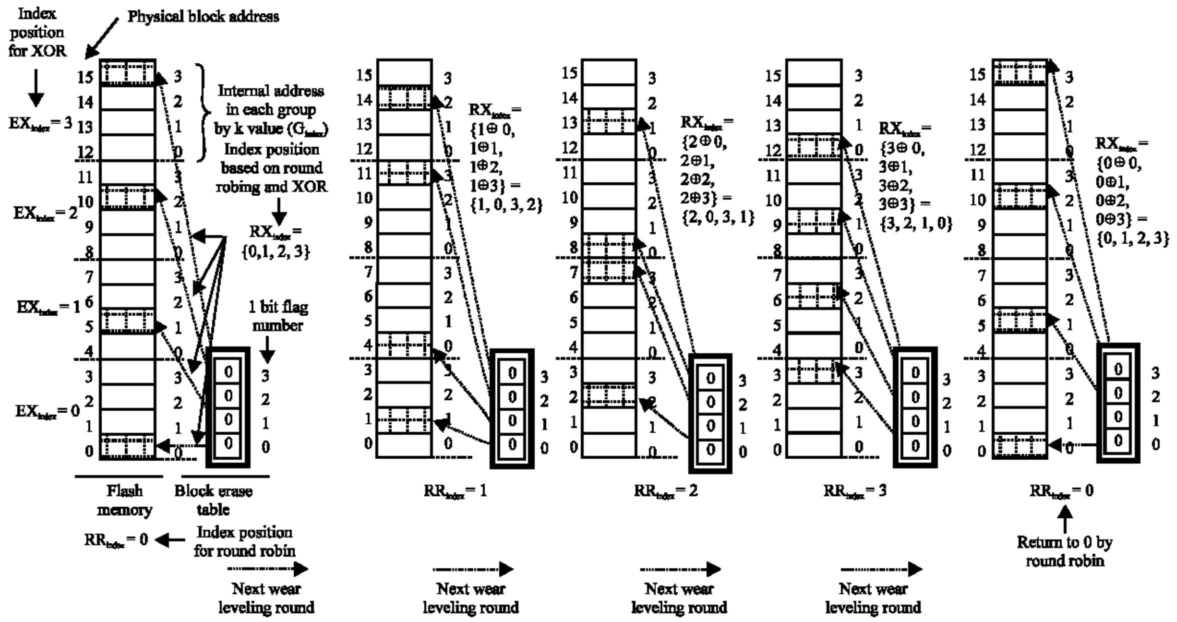
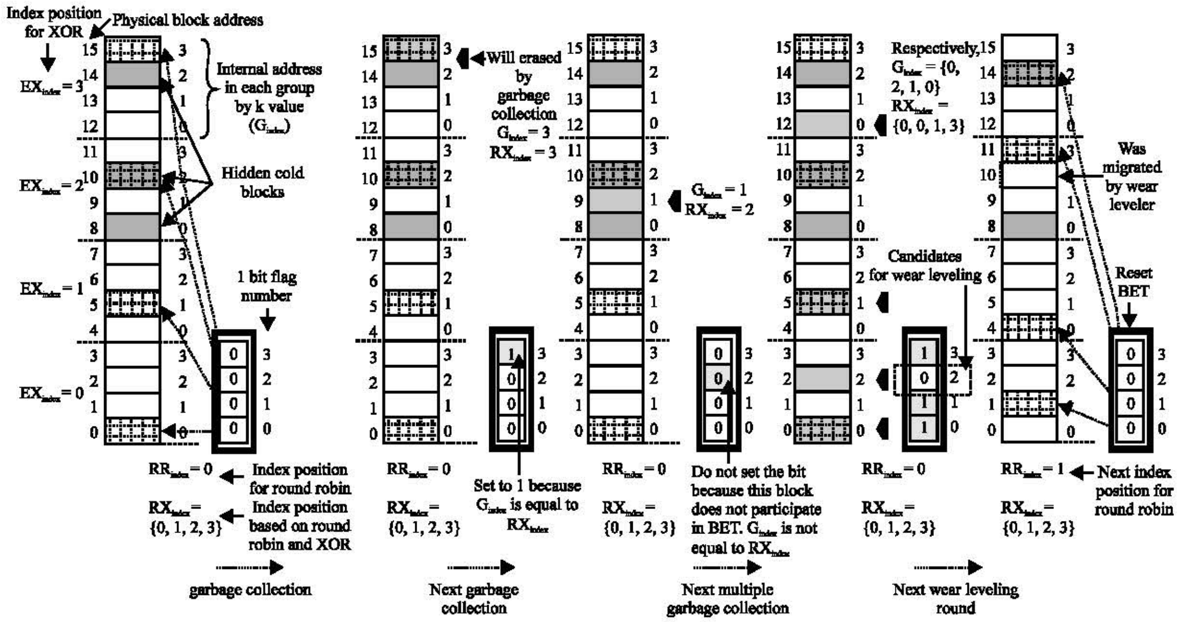

Index position for XOR
Physical block address
Internal address in each group by k value (G_{index})
Index position based on round robing and XOR
EX_{index} = 3
EX_{index} = 2
EX_{index} = 1
EX_{index} = 0
RX_{index} = {0,1, 2, 3}
1 bit flag number
Flash memory
Block erase table
RR_{index} = 0
Index position for round robin
Next wear leveling round
RX_{index} = {1⊕0, 1⊕1, 1⊕2, 1⊕3} = {1, 0, 3, 2}
RR_{index} = 1
RX_{index} = {2⊕0, 2⊕1, 2⊕2, 2⊕3} = {2, 0, 3, 1}
RR_{index} = 2
RX_{index} = {3⊕0, 3⊕1, 3⊕2, 3⊕3} = {3, 2, 1, 0}
RR_{index} = 3
RX_{index} = {0⊕0, 0⊕1, 0⊕2, 0⊕3} = {0, 1, 2, 3}
RR_{index} = 0
Return to 0 by round robin

Fig. 1: Mapping method for one block by one bit using SBET



Index position for XOR
Physical block address
Internal address in each group by k value (G_{index})
Hidden cold blocks
1 bit flag number
$EX_{index}=3$
$EX_{index}=2$
$EX_{index}=1$
$EX_{index}=0$
$RR_{index}=0$
Index position for round robin
$RX_{index}=\{0, 1, 2, 3\}$
Index position based on round robin and XOR
garbage collection
Will erased by garbage collection $G_{index}=3$ $RX_{index}=3$
Set to 1 because G_{index} is equal to RX_{index}
Next garbage collection
$G_{index}=1$ $RX_{index}=2$
Do not set the bit because this block does not participate in BET. G_{index} is not equal to RX_{index}
Next multiple garbage collection
Respectively, $G_{index}=\{0, 2, 1, 0\}$ $RX_{index}=\{0, 0, 1, 3\}$
Candidates for wear leveling
Next wear leveling round
$RR_{index}=1$
Next index position for round robin
Was migrated by wear leveler
Reset BET

Fig. 2: Example of solving hidden cold problem using SBET

to the EraseBlockSet function in line 14 and forcibly moves the relevant block. When all the bits in the BET are set to 1 through garbage collection or wear leveling in line 3, the BET is reset and the round-robin index is calculated again. The eraseblockset function performs the simple task of calculating and moving the blocks that must be moved.

Algorithm 1; Pseudo code of SWL-procedure-SBET for wear leveling:

```
Input: $e_{cnt}$, $f_{cnt}$, k, $f_{index}$, BET, $RR_{index}$ and T
Output: null
1  if $f_{cnt}$ = 0 then return
2  while $e_{cnt}/f_{cnt} \geq$ Tdo
     /*size (BET) is the number of flags in the BET*/
3    If $f_{cnt} \geq$ size (BET) then
4      $e_{cnt} \leftarrow 0$
5      $f_{cnt} \leftarrow 0$
6      $f_{index} \leftarrow$ RANDOM(0, size (BET)-1) or 0
7      reset all flags in the BET
8      $RR_{index} \leftarrow (RR_{index}+1)$ mod $2^k$;/*set next round robin
       index*/
9      return
10   end
11   while BET[$f_{index}$] = 1 do
12     $f_{index} \leftarrow (f_{index}+1)$ mod size (BET)
13   end
14     EraseBlockSet($f_{index}$, k); /*Request the cleaner to do
       garbage collection over the selected block set*/
15   $f_{index} \leftarrow (f_{index}+1)$ mod size(BET)
16 end
```

Algorithm 1; Pseudo code of SWL-SBET Update for update of BET:

```
Input: $e_{cnt}$, $f_{cnt}$, k, $b_{index}$, $RR_{index}$, $G_{index}$, $EX_{index}$, $RX_{index}$ and BET
Output: $e_{cnt}$ and $f_{cnt}$ are increased by the erased block address
$b_{index}$. BET is updated by $b_{index}$ related from k and $RR_{index}$ in
the BET mapping
1  $e_{cnt} \leftarrow e_{cnt}+1$; /*Increse the total erase count*/
2  $Ex_{index} \leftarrow [b_{index}/2^k]$ mod $2^k$
3  $Rx_{index} \leftarrow EX_{index} \oplus EX_{index}$;/*Get XOR-based round robin index*/
4  $G_{index} \leftarrow b_{index}$mod $2^k$; /*Get internal address of the group
   based on k and $b_{index}$*/
   /* If $G_{index}$ is equal to $RX_{index}$ a related bit is set in the BET*/
5  If $RX_{index} = G_{index}$ then
     /*Update the BET if needed. */
6    if BET [[$b_{index}/2^k$]] = 0 then
7      BET [[$b_{index}/2^k$]] $\leftarrow$ 1;
8      $f_{cnt} \leftarrow f_{cnt}+1$; /* increase the total Is count of BET*/
9    end
10 end
```

The SWL-SBET update algorithm in algorithm 2 shows the pseudo code that sets the relevant bits in the BET to 1 when valid pages of relevant blocks (b_{index}) are moved and the relevant blocks are erased through garbage collection or wear leveling. In lines 2-4, the XOR-based round-robin index and the internal address are converted. If these values are identical, the relevant bits in lines 6-9 are set to 1. Figure 2 shows the inclusion of cold blocks in wear leveling using SBET. The total number of blocks, value of "k" and the number and location of cold blocks are identical to that of Fig. 1.

When garbage collection occurs while the round-robin index is 0 and block 15 which is a victim block is selected and erased, block 15 takes the internal address of 3 and the XOR-based round-robin index becomes 3. As the values of the two variables are identical, the third bit in the BET is set to 1. When block 9 is selected as the victim block during the second garbage collection, its internal address will be 1 and the XOR-based round-robin index will be 2. In such cases, the BET's bit is not set to 1 and instead retains its original value because the values of the two variables are different. In the same way, if block 0, 2, 5 and 12 are selected as the victim blocks in the garbage collection, all the bits excluding the BET's second bit will be set to 1. Then, once the conditions are right for performing wear leveling, the second bit will have a BET bit value of 0. Therefore, when the round-robin index is 0 and the BET's second bit is 0, block 10 is moved because it becomes the relevant block. Using the same method, block 8 and 14 which are the remaining cold blocks are moved during subsequent wear leveling.

RESULTS AND DISCUSSION

A simulator was built to evaluate the SBET method and the results were compared against the existing BET technique and BST technique. Table 1 shows the simulation environment. Assuming that each file applied in the simulation is of the same size, the initial free space was set to 15% (Fig. 3).

Among all the files, the renewal rate was applied only to 700 files and the remaining 300 files were set with a renewal rate of 0. The cold data such as the operational data was applied. Each file was placed randomly, so that, the cold blocks and hot blocks would be stored in an irregular fashion instead of being repeatedly stored in the same physical location. In the simulation, the page-based mapping technique (FTL) and the block-based mapping technique (NFTL: NAND FTL) were used as the address mapping techniques. The MLC NAND flash memory of NFTL was considered, so that, only one replacement block would b e used for each logical block address. As FTL requires more memory in the mapping table than NFTL, NFTL is frequently used for embedded devices. However, NFTL has a slower read speed than FTL and requires more time to perform garbage collection (Ma *et al.*, 2014).

Figure 3 shows the distribution of the number of block erases for each scenario after performing garbage collection that uses the Greedy Algorithm (GA), without proper wear leveling and shows the results of performing

Table 1: Details of the simulation environment

Items	Values and description
Block count	2048
Page count per block	128
Page size	4 kB
Total capacity	1024 MB
Each file size	222 page (888 kB)
File count	1000
Garbage collection trigger	The No. of free is under 102.0 (2%)
Initial free space	15%

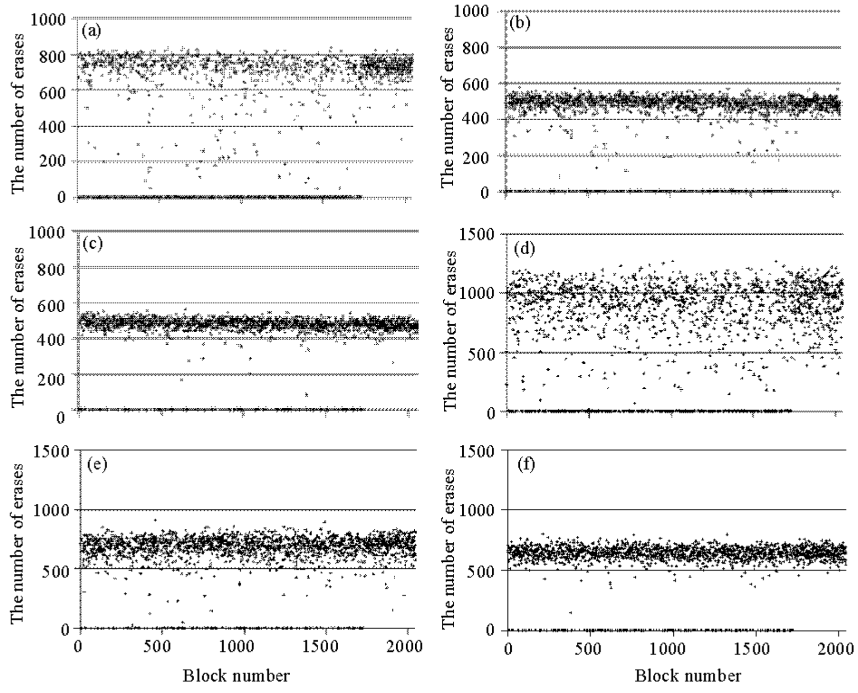


Fig. 3: Distribution of the number of block erases for each case when GA is used in FTL and NFTL: a) FTL, case 1; b) FTL, case 2; c) FTL, case 3; d) NFTL, case 1; e) NFTL, case 2 and f) NFTL, case 3

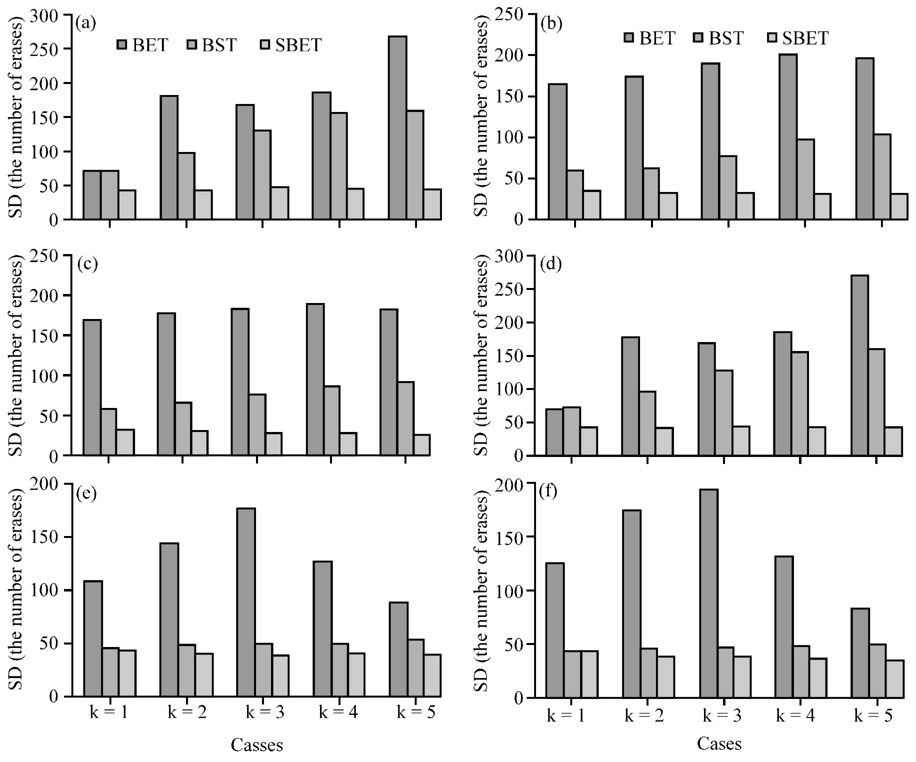


Fig. 4: Standard deviation of the number of block erases for each wear leveling technique in FTL and NFTL (T = 10): a) FTL, case 1; b) FTL, case 2; c) FTL, case 3; d) NFTL, case 1; e) NFTL, case 2 and f) NFTL, case 3

the write operation 100 million times. As NFTL requires more time for garbage collection than FTL, the average number of block erases for NFTL is higher than that of FTL. It also shows that NFTL has a greater dispersion for the number of block erases than FTL. In each of figure below, there are blocks with 0 as the number of block erases and their renewal rate is also 0. Since, an invalid page did not occur for these blocks, they were not subject to garbage collection.

Figure 3 shows the standard deviation of each technique when the value of "k" is 1 or more and "T" is 10. The standard deviation was measured when the write operation was performed 100 million times. In the FTL, BST has a lower standard deviation than BET, however, its standard deviation increases as the value of "k" increases. However, SBET shows no major difference in standard deviation even if the value of "k" increases in FTL. This shows that SBET will have the same performance even though the value of "k" increases, unlike other techniques in the FTL.

In the NFTL of Fig. 4, BET shows an irregular standard deviation regardless of the value of "k". This irregular standard deviation results from failing to recognize the cold blocks accurately which is the cause of the hidden cold block problem. As shown in b and c when the value of "k" is 5 in FTL, the BET's standard deviation decreases slightly to show an irregular form. The fact that NFTL is more irregular than the FTL implies that the number of blocks that are erased in NFTL may be double the number of blocks erased in FTL because additional replacement blocks are connected to the logical block address based on the characteristics of NFTL when garbage collection is performed. Therefore, this characteristic intensifies the irregular standard deviation according to the value of "k". However, even though the address mapping techniques differ, the standard deviation of BST and SBET increases uniformly or constantly. Unlike NFTL and FTL, BST showed a relatively balanced performance, however, SBET still had a low standard deviation. As a result, in FTL, the standard deviation of SBET decreased by up to 84% more than that of BET and decreased by up to 73% more than that of BST. In NFTL, the standard deviation of SBET decreased by up to 81% more than that of BET and by up to 27% more than that of BST.

CONCLUSION

A Sampling-based Block Erase Table (SBET) technique was proposed in this study to extend the lifespan of flash memory. For SBET, the XOR-based round-robin method was applied to the existing BET technique to minimize the performance degradation that occurs when memory usage decreases. The cause of this performance degradation was attributed to the hidden cold block problem and this was pointed out as resulting from having one bit connected to several blocks. To resolve this problem, each bit in the bit arrangement table that stores each block's erase history was configured to correspond to one block. In the experiment, the BST technique which was studied to improve the BET and the existing BET technique were implemented and compared against the proposed method. The results showed that SBET had extended the flash memory lifespan by up to 80% more than BET and up to 53% more than BST. As it had a relatively balanced lifespan extension effect compared to other techniques, even when the memory usage was reduced, it is deemed appropriate for use in sensor nodes with memory limitations or IoT devices. As the usable memory size is already defined for the SBET technique, there is a need for a method that can improve performance by increasing the maximum usable memory size in devices with limited memory resources. Hence, methods that can solve this issue will be investigated in future studies.

REFERENCES

Chung, T.S., D.J. Park, S. Park, D.H. Lee and S.W. Lee *et al*., 2009. A survey of flash translation layer. J. Syst. Archit., 55: 332-343.

Kim, S.H., J.W. Kwak and C.H. Park, 2015. Wear leveling technique using bit array and bit set threshold for flash memory. J. Korea Soc. Comput. Inf., 20: 1-8.

Kwon, S.J., A. Ranjitkar, Y.B. Ko and T.S. Chung, 2011. FTL algorithms for NAND-type flash memories. Des. Autom. Embedded Syst., 15: 191-224.

Lawton, G., 2006. Improved flash memory grows in popularity. Comput., 39: 16-18.

Lee, S. and J. Kim, 2014. Improving performance and capacity of flash storage devices by exploiting heterogeneity of MLC flash memory. IEEE. Trans. Comput., 63: 2445-2458.

Ma, D., J. Feng and G. Li, 2014. A survey of address translation technologies for flash memories. ACM. Comput. Surv. CSUR., 46: 1-39.

Park, B., S. Cho, M. Park, S. Park and Y. Lee *et al*., 2012. Challenges and limitations of NAND flash memory devices based on floating gates. Proceedings of the 2012 IEEE International Symposium on Circuits and Systems (ISCAS'12), May 20-23, 2012, IEEE, Seoul, South Korea, ISBN:978-1-4673-0218-0, pp: 420-423.

Xu, G., M. Wang and Y. Liu, 2014. Swap-aware garbage collection algorithm for NAND flash-based consumer electronics. IEEE. Trans. Consum. Electron., 60: 60-65.

Yang, M.C., Y.M. Chang, C.W. Tsao, P.C. Huang and Y.H. Chang *et al*., 2014. Garbage collection and wear leveling for flash memory: Past and future. Proceedings of the 2014 International Conference on Smart Computing (SMARTCOMP), November 3-5, 2014, IEEE, Hong Kong, China, ISBN:978-1-4799-5712-5, pp: 66-73.