A Data Recovery Technique Improves On Hybrid-Mapping For NAND Flash Memory

Van-Dai Tran and Dong-Joo Park*

Abstract: Flash memory has been built upon EEPROM (Electronically Erasable Programmable Read-Only Memory). Unlike traditional magnetic disks, Flash memory has disadvantages, like the limitations of life cycle and erase-before-write, which require a resolution well-known namely Flash Translation Layer (FTL) to resolve. Volatile memory today is used to save periodic retrieve requests for mapping-tables in flash memory. These tables can be missed when an unexpected power outage occurs. In order to address this problem, Page-mapping, Block- mapping, and Hybrid-mapping methods have been introduced. However, these methods also have shortcomings, for example, the mapping-information management overhead and the recovery time. In this paper, we introduce a data recovery scheme improves on Hybrid-mapping together with the spare area separate to ECC (Error Code Correction), block information, ASN (Allocation Sequence Number), mapping-information, Flag, and reserved in FTL. The results display that our technique has the less recovery time and mapping-information management overhead than the previous methods.

Index Terms: NAND Flash memory, Data recovery technique, ECC, Power Loss Recovery, and Spare area.

1. INTRODUCTION

Today, NAND flash memory has been more widely famous due to its well-known benefits, such as higher density, fast data speed, and lack of noise. However, it also has disadvantages, namely the limitations of life cycle, asymmetric read/write response, and erase-before-write, which require a software layer well-known namely Flash Translation Layer (FTL) to resolve. The aim of FTL consists of address translation, wear-leveling and garbage collection. Volatile memory nowadays is utilized to save periodic retrieve mapping-tables of flash memory. Nevertheless, these tables can be missed during an unexpected power outage. In order to remedy this problem, Page-mapping, Block-mapping, and Hybrid-mapping methods have been proposed. The benefits of Page-mapping are address mapping translated to logical pages and physical pages which depend on the big size of its mapping-tables. However, the Block-mapping does not depend on large mapping tables while face to problem of the effectuation deterioration because of the lack of ability for replaces the flash memory systems. So, to resolve the drawbacks of Page-mapping and Block-mapping, Hybrid-mapping has been commonly deployed. The Page-mapping method using spare-area for stores the mapping-information of every page. The mapping-information would be saved in the spare area while the pages writing execution, thus no extra actions to manage the mapping-information. Consequently, the drawback of the recovery time can be extended for the requirement to retrieve all pages of the flash memory. For decrease the recovery time of the Page-mapping, the Block-mapping technique utilizes the spare-area to save the mapping-information of page. In addition the mapping-table is regularly saved into the map blocks of the flash memory. The lack of ability of overwriting flash memory results in the entire map block to rest unaltered accompanied by the extra mapping- information inserted in each new page updated. So we call this initial map block the basis map block, also its pages, the basis map pages. The addition pages are called as appended map pages. While an empty block is refilled the appended mapping pages, new map blocks with available mapping-information can be deployed by a merge execution. If a power failure occurs, the stored mapping-table of map blocks requires reading, so the recovery time will be faster. Nevertheless, the Block-mapping technique needs added write, merge, and erase executions that limit the flash memory effectuation to upgrade map blocks. However, these methods still have shortcomings such as the recovery time and the mapping-information management cost. In this paper, we introduce a data recovery technique improves on Hybrid-mapping with the spare area separated ECC, block information, ASN, mapping-information, Flag, and reserved in FTL with the shorter recovery time and smaller mapping-information management cost than the previous methods. The rest of paper organizes as follows. Section 2 indicates the literature and relevant works, namely FTL, Power Loss Recovery (PLR) techniques, and ECC. Section 3 presents our technique and comparisons with other methods. And finally the conclusion of the paper is shown in Section 4.

- *Van-Dai Tran is currently a Ph.D. candidate in Department of Computer Science and Engineering at Soongsil University, Seoul, Korea (06978). E-mail: tranvandai@soongsil.ac.kr*
- **Corresponding author Dong-Joo Park is currently a Professor in School of Computer Science and Engineering at Soongsil University, Seoul, Korea (06978). E-mail: djpark@ssu.ac.kr*

2 BACKGROUND AND RELATED WORKS

2.1 Flash memory

Flash memory [1-2] has been widely developed today as nonvolatile storage. In contrast to regular magnetic disk, the flash memory includes an array of memories, controllers together RAMs which utilized as input buffers and save connection information. In addition, flash memory includes some fundamental operations such as reading, writing, and erasing. Flash memory needs a solution well-known namely FTL [1-2] for manipulate also organize its data to enhance the effectuation of flash memory. Nowadays, NOR and NAND are popular of flash memory are used. While NOR flash memory is widely utilized into the factories with many characteristics are using easily and usable interface for coding performance as a well device unaccompanied by data storage. NAND flash memory meets capacity, store data smaller and low-priced. Nevertheless, engineers are no attracted for utilize it due to compound organization and abnormal its interface. Table 1 depicts a comparison between NOR flash and NAND flash and fig. 1 describes the structure of NAND Flash memory.

Table 1
Comparison of NOR flash and NAND flash [15]

Feature	NOR flash	NAND flash
Capacity	8MB-256MB	256MB-2GB
Cost per bit	Higher	Lower
Random read speed	Faster	Slower
Write speed	Slower	Faster
Erase speed	Slower	Faster
Power on current	Higher	Lower
Standby current	Lower	Higher
Bit-flipping	Less common	More common
Bad block development	Less frequent	More frequent
Bad block handling	Not mandatory	Mandatory
Data retention	Very high	Lower
Program-erase cycles	Lower	Higher
Preferred application	Code storage and execution	Data storage

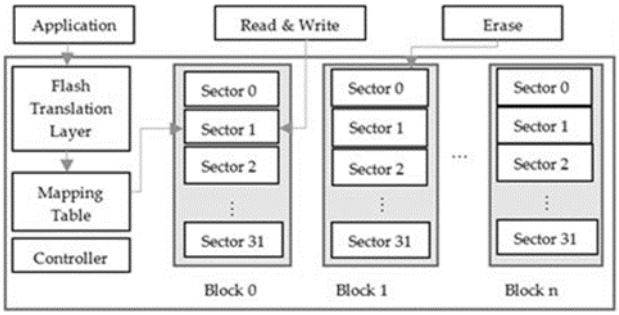

Fig. 1 *Organization of NAND Flash Memory*

2.2 FTL

FTL's [1] common functionalities are logical-to-physical address conversion, wear leveling, and garbage collection. In FTL mapping algorithms are separated different schemes for example Page-mapping, Block-mapping, and Hybrid-mapping. The Page-mapping utilizes logical pages matching to all physical pages, so it requires larger mapping tables. The block mapping handles the block like a mapping unit and has the problem of the effectuation deterioration because of the lack of ability for overwriting flash memory system. Thus, Hybrid-mapping is utilized resolve the drawbacks of Page-mapping and Block-mapping with the high effectuation of FTL.

2.3. PLR (Power Loss Recovery)

Flash memory is hardware constitutional which is not able to stay away from the unexpected power loss [2-3]. In flash memory, the power failure happen, it might subject to undesirable errors. Therefore, management of the power loss in the flash memory is essential. If the mapping-table is exchanged from the page-mapping of FTL, the data will be updated or erased. While the data is updated, flash memory writes data to other address as it is not to be overwritten. Consequently, the mapping-table should be revived to return the exchanged address. The updated data could not be retrieved, if not revised. After that, as the mapping-table is revised, the data will be erased from flash memory. If the data is deleted from the controller or the garbage collection of the FTL, the mapping-table will removes address information from the deleted data. While the mapping-table is updated, it stores the exchanged information from flash memory to restore if the power loss happens after that. Nonetheless, saving the mapping- information of flash memory when the data is updated leads to effectuation degradation and capacity insufficiency. So, the FTL executes a data backup operation known as flush to discover a resolution for the problem mentioned above. FTL creates flush procedures regularly and saves mapping-information in data area or spare area of the flash memory [2-3].

2.3.1. In-Page Backup

This technique using the spare area of pages for saves the backup data. In general, spare area would be saved the management information of FTL also the ECC bits [7]. However, nowadays with the technics developed the page size of flash memory has been risen due to the size of spare area also has been risen. Therefore, the unsuitability area in spare area can be allocated to write the backup data, that is, a mapping information [2-3]. While this technique is operated in flash memory, it is not able incur extra cost. In flash memory, recovery process operates, while a power loss occurs. So, the recovery process as follows. Firstly, it reads every spare area of pages in flash memory. Second, it constructs the mapping tables from the read information. Although the process is very simple, the recovery postpones very long.

2.3.2. In-Block Backup

This technique saves the backup data in map blocks, a number of preserved blocks in flash memory. Since the mapping-information is created, this technique saves it in map-blocks [2]. So, this technique creates the added writing operations, it will be spent extra cost. If the power loss in flash memory occurs, recovery process will browse map-blocks. In addition it constructs the mapping tables from the read mapping-information. Unlike the In-Page Backup recovery technique, the recovery postpones in In-Block Backup is smaller.

2.3.3. Hybrid Backup

This technique utilizes the mixture of both methods are referred above and lessen both recovery time and overhead of running backups. In addition, page-mapping updates the logical page number (LPN) of every page writing, which is retained from the backup area of the matching pages, counts the page of mapping-table, and updates the block-mapping like logical block number (LBN), which is saved in the block map. Through the recovery procedure, page-mapping and block-mapping are recovered from the map blocks and superfluous areas of the common pages. This technique increases retrieval through In-Block Backup and reduces recovery time measured to In-Page Backup. The runtime backup overhead of this technique is as smaller than the block backup as the read/write latency of the backup zone able is insignificant. Table 2 depicts the comparison of advantages and disadvantages of In-Page, In-Block, and Hybrid Backup technique.

Table 2
A comparison of In-Page, In-Block and Hybrid

	In-Page	In-Block	Hybrid
Advantage	Comes up with high performance and lifetime	Decreases translation table size [11]	Reduces erase-before-write problem compare with

	[9-10]		the page-mapping table [8, 12]
Disadvantage	Needs big address-table conversion which is saved in RAM and it able be less density; most of area cost is raised.	The maintaining countervails both logical and physical block are similar, on writing a page of the block, makes every available pages of the block able to duplicate from an available block.	Needs maintaining block and page-mapping tables, due to greater saving cost. Block-merge execution dues to more delete and writing executions.

2.3.4. A-PLR (Accumulation for Power Loss Recovery)

This technique is introduced by Jung et al. [3] which is based on advanced In-Page Backup scheme. It keeps the mapping-information in the spare area like In-Page Backup. Different from In-Page Backup, mapping-information of A-PLR is stored by an unusual approach, that able resolve the shortcoming of the In-Page Backup. The process is as follows. This scheme utilizes a buffer into its RAM, called MIB (Mapping Information Buffer), in which size is stable, also the place which the gathering of the mapping-information able is saved. As A-PLR stores mapping-information, the amount of the refilled slots inside MIB rises. If MIB is full, it able saves the mapping-information in the intermediate page in the spare area. While the recovery process of a power loss occurs, A-PLR reads the intermediate page. So, the recovery latency is smaller than In-Page Backup as A-PLR using a re-mapping-table to read only intermediate pages.

2.3.5. HYFLUR (HYbrid FLUsh Recovery)

This scheme has been developed from the Page-mapping approach. HYFLUR [4] is performed by using the Open SSD Project Board [17] and the platform with 64GB NAND flash make up of 8 KB pages memory and memory blocks with 128 pages. In short, NAND memory contains 66,432 blocks in total and uses the mobile 64 MB SDRAM [18]. This scheme saves the specific blocks of flash memory in RAM to keep all mapping information by using the 15 blocks MSB (Mapping information Store Block) and 25 blocks TSB (Table Store Block). Thus, this scheme saves 40 blocks, and space cost is about 0.06%. Thus the cost is not inconsiderable [4].

2.3.6. C-HYFLUR (Compression algorithm integrated into HYbrid FLUsh Recovery)

This scheme uses the compression algorithm integrated into the recovery algorithm HYFLUR [4]. The LZSS compression algorithm is improvement of LZ77 [6], and it is used for the HYFLUR algorithm to overcome the space cost that is a drawback of the page-mapping technique. C-HYFLUR is performed in the specific device similar to HYFLUR. The rate of compression of LZSS is approximately 31% and 8K of data can be compressed to 2.6K. So the C-HYFLUR can save three times more mapping information in MSB than HYFLUR and uses only 9 TSB blocks, a large decrease from 25 blocks needed by HYFLUR [5]. In short, A-PLR retrieves all most of the intermediate pages from the recovery process. Therefore, the quantity of pages to read is spent more than HYFLUR. The compression algorithm reduces the number of a reading operation and solving the space cost of the page mapping of the FTL. Thus, the recovery time cost of C-HYFLUR is lesser than A-PLR and HYFLUR. Hence, the append page writing operations and compression process are necessary; the response time is larger than A-PLR and HYFLUR [19].

2.4. ECC

The aim of the ECC is to append an additional condition to messages for the event when a reading might discover errors and restore all messages which are written. The block error code correction relies on data sub-sectors. And depending on the error fixing outline used; it utilizes a different amount of redundancies which is known as the parity bit. In the inequality of Hamming algorithm the lowest parity bit can be calculated as shown-in (1), where n is the length of code words, k is bits for information, and t is bits for ECC.

(1)

$$\sum_{i=0}^{t} \binom{n}{i} \leq 2^{n-k}$$

The size of sub-sector is lower than the square area size. In order to address the problem of collision of the flash die size, BCH (Bose-Chaudhuri-Hocquenghem) and Reed Solomon [12] have been proposed, but Reed Solomon code needs more bits than the other. So, that is the reason why flash memory adopts BCH to ECC [13].

3 PROPOSED TECHNIQUES

Our technique is based on Hybrid mapping and using the spare area separated ECC, block information, ASN, mapping-information, Flag, and reserved. Every page owns a LPN for the page address restoration. The controller browses every first page for every block sorted in descending order by ASNs to build the mapping table. If the flag is equal to 1, the block is free. Otherwise, the block has a programmed page. In the recovery time, the FTL may lessen the amount of reading operations from blocks and save the invalidate page of the block named physical block number (PBN). Fig. 2 shows the organization of the spare area in our scheme and fig. 3 depicts the data recovery process.

ECC	Block information	ASN	Mapping information	Flag	Reserved

Fig. *2 Structure of the spare area*

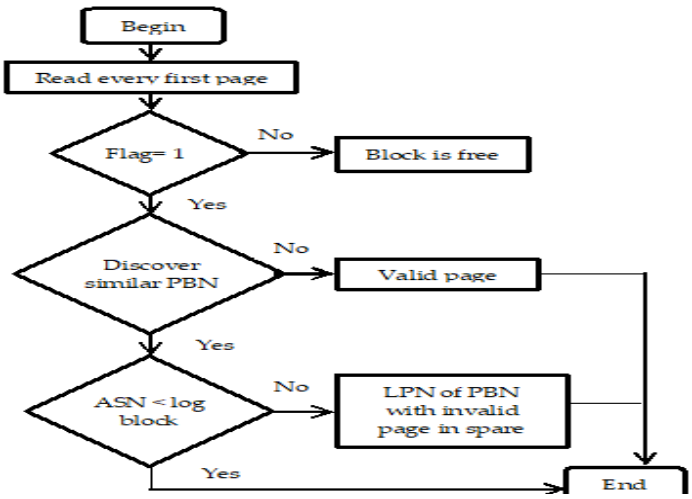

Fig. 3 *The process of data recovery*

The process starts by the controller reading each first page of every block to build the mapping table through the system reboot and discover whether the data block containing the same LPN page of block. If not, the mapping-table is upgraded from the page of log block as an available page. And then the controller compares the ASN of data block with the log block, and if the ASN of data block is larger than log block, the mapping-table is upgraded from the page of data block for correlating with LPN of PBN into spare area. The PBN presents the block address consisting of invalid pages. To implement the proposed scheme, we compared the read time and recovery time of our method with Page-mapping, Block-mapping and Hybrid-mapping scheme. In this paper, for the assessment of the recovery time, we assume the simulation device has the following parameters: 4096 bytes pages, 128 bytes spare area, 128 pages block, also using the BCH algorithm in which 4 bits ECC in 1024 bytes concern the NAND flash datasheet [16] and 63 bytes inside the space area able be assigned for ECC [7]. The device simulation of this paper is using a 2 GB address space for a 4KB page unit and 10 log blocks. Table 3 summarizes the technique parameters of NAND flash definition utilize the assessment of recovery time.

Table 3
NAND Flash Specification [18]

Parameter	Symbol	NAND flash spec.
Page size	S_P	4096 bytes
Pages per block	N_P	128 pages
Number of blocks	N_B	4096 blocks
Capacity	C	2GB
Data transfer time from cell to register	t_D	60 µs
Program time	t_P	800 µs
Block erase time	t_B	1.5 ms
Write/ Read time	t_W / t_R	25 ns
Spare page size	S_{SP}	128 bytes

We assess the read and the recovery operation time for each technique as follows.

1) For the Page-mapping technique:

The mapping-information inside the space areas has 4 bytes, so the reading execution time is t_{R1}

$$t_{R1} = t_D + (t_R \times 4) = 60.1\ (\mu s)$$

A block map table accomplished time is t_{B1}

$$t_{B1} = t_{R1} \times 128 = 7{,}692\ (\mu s)$$

Device has 4096 blocks, and whole recovery time is t_{rec1}

$$t_{rec1} = t_{B1} \times 4096 = 31{,}510\ (ms)$$

2) For the Block-mapping technique:

Reading execution time is t_{R2}

$$t_{R2} = t_D + (t_R \times 4096) = 162.4\ (\mu s)$$

Whole recovery time is t_{rec2}

$$t_{rec2} = t_{R2} \times 128 \times 4 = 83.15\ (ms)$$

3) For Hybrid-mapping technique:

Reading execution time is t_{R3}

$$t_{R3} = t_D + (t_R \times 4096) \times 4 = 469.6\ (\mu s)$$

Whole recovery time is t_{rec3}

$$t_{rec3} = t_{R3} + (t_{R3} \times 128) / 2 = 30.524\ (ms)$$

4) For our technique:

The number of log blocks is 10, the controller read the first page of each blocks for mapping table, so the reading operation time is t_{R4}

$$t_{R4} = t_D + (t_R \times 4096 \times 10) / 2 = 572\ (\mu s)$$

The controller browses each first pages of every block for build the mapping table. In addition, the PBN shows the block address and lack of valid pages is 4 bytes, and we assess the average based on the number of saved data. Thus, the whole recovery time is t_{rec4}

$$t_{rec4} = t_{R4} + ((t_D + (t_R \times 4)) \times 128 \times 10) / 4 = 19.804\ (ms)$$

5) Comparison:

We assess our scheme with Page-mapping method, Block-mapping, and Hybrid-mapping method which has more efficient time of power-loss recovery. The results show that our scheme are as well as Hybrid-mapping in the reading time and ambitious in the recovery time, as shown in Fig. 4 and Fig. 5.

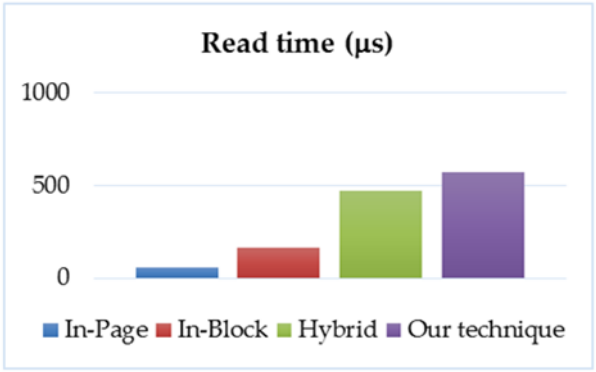

Fig. 4 *Read time comparison*

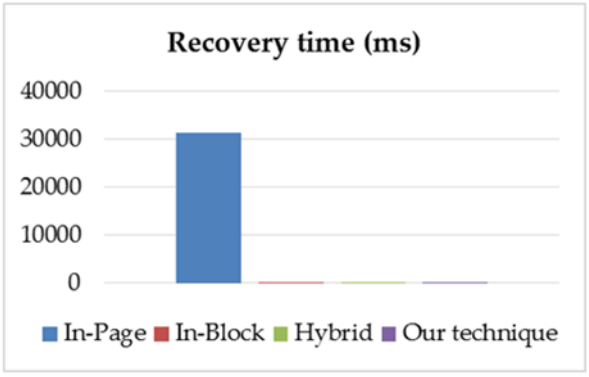

Fig. 5 *Recovery time comparison*

4 CONCLUSION

In this paper, we proposed the scheme improved Hybrid mapping using the spare area in FTL. This technique reduces the map-block management costs and less recovery time, which enhances the effectuation of NAND flash memory. In near future, our target is embedding the data recovery algorithm for the flash-based applications and to investigate the productiveness of recovery effectuation.

ACKNOWLEDGMENT

This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the National Program for Excellence in SW) (2018-0-00209) supervised by the IITP (Institute of Information & communications Technology

Planning & Evaluation)

REFERENCES

[1] T.-S. Chung, D.-J. Park, S. Park, D.-H. Lee, S.-W. Lee and H.-J. Song, “A Survey of Flash Translation Layer”, Journal of Systems Architecture, vol. 5-6, no. 55, (2009), pp. 332-343.

[2] T-S. Chung, M. Lee, Y. Ryu, and K. Lee, “PORCE: An efficient power off recovery scheme for flash memory”, Journal of Systems Architecture, vol. 10, no. 54, (2008), pp. 935-943.

[3] S. Jung and YH. Song, “Data loss recovery for power failure in flash memory storage systems”, Journal of Systems Architecture, vol. 1, no. 61, (2015), pp. 12-27.

[4] J.-H. Chung and T.-S. Chung, “HYFLUR: recovery for power-off failure in flash memory storage systems using HYbrid FLUsh recovery”, Information science and applications (ICISA 2016), no. 376, (2016), pp. 501-510.

[5] J.-H. Chung, S. Kim and T.-S. Chung, “C-HYFLUR: Recovery for Power-off Failure in Flash Memory Storage Systems Using Compression Scheme for HYbrid FLUsh Recovery”, International Conference on Mobile and Wireless Technology (ICMWT 2017): Mobile and Wireless Technologies, (2017), pp. 284-294.

[6] J. Ziv and A. Lempel, “A universal algorithm for sequential data compression”, IEEE Transactions on Information Theory 23, vol. 3, no. 23, (1977), pp. 337-343.

[7] S. Jung, S. Lee, H. Jung and YH. Song, “In-page management of error correction code for MLC flash storage systems”, IEEE Trans. Consum. Electron, vol. 2, no. 56, (2010), pp. 339-347.

[8] BW. Nam, GJ. Na, SW. Lee, “A hybrid flash memory ssd scheme for enterprise database applications”, International Asia-Pacific Web Conference (APWEB), IEEE, (2010), pp. 39–44.

[9] S. Im, D. Shin, “Comboftl: Improving performance and lifespan of mlc flash memory using slc flash buffer”, Journal of Systems Architecture 2010, vol. 12, no. 56, (2010), pp. 641–653.

[10] Y. Oh, E. Lee, J. Choi, D. Lee, SH. Noh, “Hybrid solid state drives for improved performance and enhanced lifetime”, Symposium on Mass Storage Systems and Technologies (MSST), (2013), pp. 1–5.

[11] R. Chen, Z. Qin, Y. Wang, D. Liu, Z. Shao, Y. Guan, “On-demand block-level address mapping in large-scale nand flash storage systems”, IEEE Transactions on Computers 2015, vol. 6, no. 64, (2015), pp. 1729–1741.

[12] S. Im, D. Shin, “Storage architecture and software support for slc/mlc combined flash memory”, Proceedings of the 2009 ACM symposium on Applied Computing, (2009), pp. 1664–1669.

[13] R. Micheloni, A. Marelli and R. Ravasio, “Error Correction Codes for Non-Volatile Memories”, Springer-Verlag, (2008).

[14] R. Micheloni, R. Ravasio, A. Marelli, E. Alice, V. Altieri, A. Bovino, L. Crippa, E. Di Martino, L. D'Onofrio, A. Gambardella, E. Grillea, G. Guerra, D. Kim, C. Missiroli, I. Motta, A. Prisco, G. Ragone, M. Romano, M. Sangalli, P. Sauro, M. Scotti and S. Won. “A 4Gb 2b/cell NAND Flash Memory with Embedded 5b BCH ECC for 36MB/s System Read Throughput”, IEEE International Solid-State Circuits Conference Dig. Tech. Papers, (2006), pp. 142-143.

[15] Avinash Aravindan, “Flash 101: NAND Flash vs NOR Flash”, Available online: https://www.embedded.com /flash-101-nand-flash-vs-nor-flash. (Accessed on September, 2019).

[16] Samsung Electronics, K9LCG08U1A Data sheet online. Available online: http://www.datasheet-pdf.com/ PDF/ K9LCG08U1A-Datasheet-Sam-sung-704175. (Accessed on September, 2019).

[17] Indilinx Jasmine Platform Specification. Available online: http://www.openssd-project.org/wiki/ Jasmine_OpenSSD_Platform. (Accessed on September, 2019).

[18] Samsung Electronics, K4M51323LC Data sheet online. Available online: http://www.datasheet-pdf.com/PDF/K4M51323LC-F-Datasheet-Samsungsemiconductor-659502. (Accessed on September, 2019).

[19] [19] V.-D. Tran, D.-J. Park, A survey of data recovery on flash memory, International Journal of Electrical and Computer Engineering, (2020), 360-376, 10(1).