A Compression Layer for NAND Type Flash Memory Systems

Wen-Tzeng Huang, Chun-Ta Chen, Yen-Sheng Chen, and Chin-Hsing Chen
National Taipei University of Technology, ROC
{wthuang,s3419002,s2418002}@ntut.edu.tw; chchen@ctc.edu.tw

Abstract

Storage devices of embedded systems must have the characteristics of small size, great capacity, low-power consumption, lightweight, non-volatility, and vibration resistance. The NAND type flash memory, briefly denoted by NandFlash, is one of the more often-used storage devices. In terms of unit price, its cost is several dozen to hundred times more expensive than the traditional Hard-Disk (HD), since its storage space is limited. Therefore, to increase the storage space of NandFlash is great significance. In this paper, we improved the compression layer for NandFlash, which can be coordinated with the X-RL algorithm, to avoid overhead and reduce the degree of internal fragmentation in the compressed data pages. Hence, our proposed method can improve the compression rate. In the reading phase, we use the consecutive memory allocation method, which can reduce the superfluous time caused by non-consecutive access. Therefore, our architecture is meaningful and practical for embedded system applications.

Index Terms — NandFlash, Embedded system, Compression Algorithm, X-RL.

1. Introduction

With technological advances, many electronics portable products, such as PDA, cellular phone, DSC, MP3, and digital voice recorder, must use non-volatile memory. Flash memory can be used to meet these portability requirements. As a result, SmartMedia, CF, and SD cards have recently become essential equipment for these portable products.

There are two types of flash memory, NOR type flash memory denoted by NorFlash and NandFlash. Since NorFlash uses random-access to access its content, it is more suitable for storing the program code. In other words, NorFlash can store the kernel of embedded systems. However, its unit cost is more expensive than NandFlash and NorFlash does not store as much data. In contrast to NorFlash, NandFlash can store large amounts of data. The unit price of NandFlash cost is several dozen and hundred times more expensive than the traditional HD, since its storage space is limited. In addition to the price issue, there are two problems with NandFlash. The writing bandwidth is limited since its writing unit is a page and it needs an erasing operation before this writing action; the storage capacity is also limited. These two problems can be solved by effective use of the compression algorithms.

NandFlash only supports the page I/O operations under the general flash memory management structure [2]-[7]. When the amounts of compression data are smaller than the page size, internal fragmentation occurs [9]. In other words, since the I/O page size is fixed, there is no compression result if this compression size is smaller than the I/O page size. Moreover, it seriously degrades the compression effectiveness. Therefore, if we want to achieve better compression results, a flash compression layer should be added into the flash translation layer in the traditional FFS management [9].

Although some compression methods are studied in the literature, their weaknesses are that they demand many buffers and also need to reduce its reading speed [9]. Therefore, increasing the usage space and reading the consecutive blocks of NandFlash merit further investigation. In this paper, we improve the disadvantages of NandFlash and then increase its practical value according to the characteristics of the X-RL algorithm and allocation strategy.

The remainder of this paper is organized as follows. Section 2 describes the working principle of flash and problem definition. The compression algorithm is discussed in Section 3. In Section 4, we present our main results. Section V contains our conclusions.

2. Basic Concepts

The standard constructions of flashes and some features of NandFlash have been discussed in the literature [2]-[8][11]-[14]. An example structure of NandFlash is shown in Figure 1. According to the application functions, the flashes can be divided into the code and data flash. In contrast to NorFlash, NandFlash has the smaller size, higher capacity, and inexpensive. The storage unit is a page and a block is composed of several pages in NandFlash. Hence, its access unit is a page and the cleaning unit is a block. NandFlash reads the data by a

Proceedings of the Third International Conference on Information Technology and Applications (ICITA'05)
0-7695-2316-1/05 $20.00 © 2005 IEEE

sequence, which uses only an 8-bit I/O port to access the data of a page unit. Therefore, in the reading data or cleaning files, especially for very large files, the access speed of NandFlash is faster than that of NorFlash. The lower random-access speed is the major disadvantage of NandFlash, since it uses a page unit to access data.

The reading data of NandFlash have a page (512B) as its unit, while an erasing operation has a block (16KB) as its unit. There are two functions, the fast writing and erasing operations in NandFlash. The erasing time of a block in NandFlash is about 2ms [11]-[14] or several hundred ms in NorFlash. Since the data and address bus is the same bus used to implement the sequential reading, the random-access data are slower and it cannot use the byte as the randomly writing method. The bit cost is the lowest for solid-state memory due to its small chip size and small number of pins. In addition, there is an invalid area, which can reach 1% of the total flash. This invalid area must be masked to avoid using it again. Generally, the size is 512B in addition to 16B of backup area in a NandFlash as shown in Table 1. That is, there is 528B in it and a block is composed of 32 pages, which is 16KB.

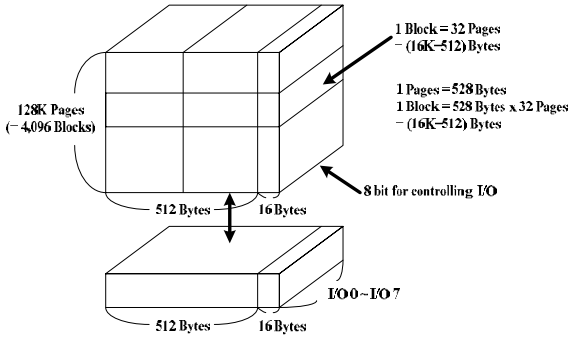

Figure 1. Structure of NandFlash.

Table 1. Specification characteristics of NandFalsh.

NandFlash	Characteristic
Page	(512 + 16)B
Page Program	(512 + 16)B
Block Erase	(16K + 512)B
Random Access	12us(Max.)
Serial Page Access	50ns(Min.)
Program Time	200us(Typ.)
Block Erasing Time	2ms(Typ.)
Endurance	100K Program/Erase Cycles
Data Retention	10 Years

NandFlash cannot be written or cleaned by a word or a byte. The cleaning unit of NandFlash is a block or the whole slice. Before we re-program codes in the same place, we must clean the blocks and the whole slice once. Since the cleaning and writing is slower and the block is larger, the cleaning and writing time will be longer. Another limit of NandFlash is the long erasing time [11]-[14]. Various management algorithms of traditional HD cannot be applied to NandFlash for these two reasons. Therefore, a NandFlash file management system plays an important role and there are some preliminary results in the literature [6].

3. X-RL Compression Algorithm

The advantages of compression data are to save the storage pace, reduce the storage or transmission cost, and increase the security. However, an appropriate method for the particular applications must be selected, although there are many compression methods. The compression method is proposed in the literature [8]-[10], which is similar to the dictionary-based model, denoted by the X-RL compression algorithm. This method can be employed in SmartMedia card applications [9]. X-RL has especially a good compression ratio for small units. This method can be designed by the simple hardware to increase the processing speed. Moreover, there is no delay time as in writing data into SmartMedia cards [11]. An example explains the compression method of the dictionary-based model as shown in Figure 2. First, this dictionary-based model will be set for a basic database. This basic database is namely “a”, “b”, and “c”; this database will increase its words as the amount of input data is progressively increased. For example, words such as “ab”, “ba”, “aba”, “abaa”, and “abac” will be added into the database.

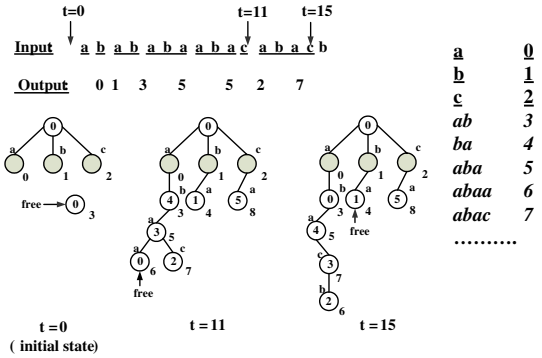

Figure 2. Example of dictionary-based compression algorithm.

Then, if one input word is “abababaabacabacb…”, compression algorithm will encodes it as “0135527…” according to the recorded phrases of the database. Thus, this can reduce the amount of data in order to achieve

Proceedings of the Third International Conference on Information Technology and Applications (ICITA’05)
0-7695-2316-1/05 $20.00 © 2005 IEEE

compression, and then through the recorded phrase of database, the encoded information can be recovered into the original information. From some simulations and experiments, the database with 128 entries is the optimum of the compression ratio [9]. However, the characteristic applications of NandFlash compression method emphasize its compression speed and lossless compression method. This process of compressing a file is such that, after being compressed and expanded, it completely matches its original data format bit-by-bit. Text, code, and numeric data files must be compressed using a lossless method; and such methods can typically reduce a file to 40 percent of its original size. For the lossless compression methods, there are two typical models. One is the statistical model, such as Huffman coding; the other is the dictionary-based model, such as LZW algorithm. To get the real-time compression speed, we adopt the above dictionary-based model, the X-RL lossless compression method, to apply in NandFlash applications.

4. Main result

First, through the basic characteristics of NandFlash, we know that the access unit is a page. Therefore, the internal fragmentation of un-used space will be reduced in the programming. In the literature [9], Yim et al. propose the Internal Packing Scheme (IPS) with the Best-Fit method for SmartMedia card applications, briefly denoted as $IPS_{\{Best\text{-}Fit\}}$ [9], whose structure is shown in Figure 3. They employ the Best-Fit method with the external buffers in this IPS strategy; and the average data page of the internal fragmentation of un-used space will be the minimum in various program file formats of applications [9].

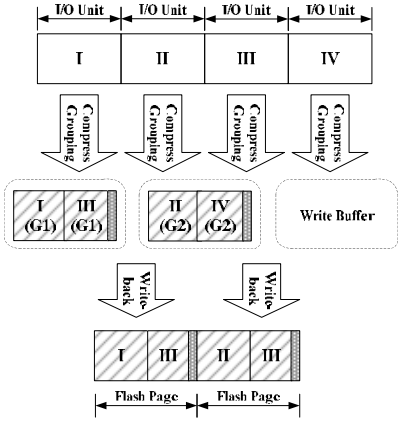

Figure 3. Internal packing scheme with the Best-Fit method.

From the compression information between NandFlash and SmartMedia cards, the specification is shown in Table 1. We learn that some subjects can be further studied. Therefore, we propose the synthesis mechanism here, which can be more perfectly applied to various applications. About the aspects of reading bandwidth, the reading time of the consecutive data is about 50ns for each byte in NandFlash, whose reading time is the high speed and efficacy. Hence, the benefit of NandFlash is applied to the continuous reading data. However, it is assigned the optimum allocation according to the compressed size in the Best-Fit method. Such allocation will present different results according to partition pages of the file types, and the consecutive characteristic of a file will become scatter. This will reduce the reading bandwidth of the system.

For example, if each file size is 2.4KB, the unit page I/O is divided into A1, A2, A3, A4, and A5; and after compressing these files, they become A1' , A2' , A3' , A4' , and A5', respectively; the allocation result of $IPS_{\{Best\text{-}Fit\}}$ is shown in Figure 4.

If the reading time of $IPS_{\{Best\text{-}Fit\}}$ method is considered the major issue, each data page must cost one reading initial latency and one reading time, since each data page is in the un-continuous allocation status. Therefore, the total cost is [(12us*5+(A1'+A2'+A3'+A4'+A5') *50ns)=60us+(A1'+A2'+A3'+A4'+A5')*50ns in this example as shown in Figure 4. It costs five initial latency times for the un-continuous reading method.

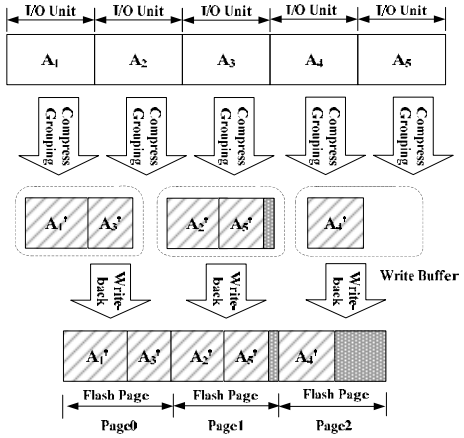

Figure 4. Sketch figure for a writing file of $IPS_{\{Best\text{-}Fit\}}$.

The allocation strategy of $IPS_{\{Best\text{-}Fit\}}$ must save the compressed page by the Best-Fit space. Moreover, when it reduces the buffer space, the data page with the minimum fragmentation rate of buffers is written into the data page of NandFlash. Such behavior is similar to the probability

combination. Therefore, the probability of the consecutive arrangement is $p=1/n!$, where n is an integer [15]. The file with the different number of pages has its average initial latency as shown in Table 2. For example, if a page with the row order is 12435967 8, the total reading initial latency times of this file is six times, 12, 4, 3, 5, 9, and 678, respectively. It also needs six times of the reading initial latency.

Table 2. Reading initial latency and average times of different size file.

		Allocation Frequency											
		1	2	3	4	5	6	7	8	9	10	Average	Average/Pages
Size	1 Page	1										1	100.00%
	2 Pages	1	1									1.5	75.00%
	3 Pages	1	2	3								2.3333	77.78%
	4 Pages	1	3	9	11							3.25	81.25%
	5 Pages	1	4	18	44	53						4.2	84.00%
	6 Pages	1	5	30	110	265	309					5.1667	86.11%
	7 Pages	1	6	45	220	795	1854	2119				6.1429	87.76%
	8 Pages	1	7	63	385	1855	6489	14833	16687			7.125	89.06%
	9 Pages	1	8	84	616	3710	17304	59332	133496	148329		8.1111	90.12%
	10 Pages	1	9	108	924	6678	38934	177996	600732	1334961	1468457	9.1	91.00%

The different size files with their reading initial latency and average times are shown in Table 2. This table shows that the average times is greater than 0.75*total number of pages; the average times will increase following by number of pages. When the file size is 10 pages, the ratio between the number of pages and its average reading initial latency time is 91%. The probability of the initial latency times and ratio figure in a file with 10 pages is shown in Figure 5. Therefore, from Table 2 and Figure 5, we can find that the reading cost of $IPS_{\{Best\text{-}Fit\}}$ increases rapidly for serious dispersion of the file.

From NandFlash characteristics, each data block must be set to the initial latency again. In other words, a 30K compression file with the consecution arrangement data type needs two initial latency times ($\lceil 30k/16k \rceil = 2$). The problem of the probability rank is a hierarchical issue. For 16K blocks, if the compression ratio is 40.8%, it can store 45 compressed pages at most [9]. Hence, the combination possibility is 45!. To computerize such a number is beyond the extreme limit computation power of a general computer. However, the relation between the number of pages and average initial latency time is almost a linear function when the number of pages is greater than 4 as shown in Figure 6. Therefore, this almost linear formula is as follows:

$$aveg\_num = \begin{cases} 1 & , \quad y=1 \\ 1.5 & , \quad y=2 \\ 14/6 & , \quad y=3 \\ 0.975y-0.65 & , \quad 32 \geq y \geq 4 \end{cases}$$

Let *aveg_num* be the average initial latency time and "*y*" be the number of pages.

Table 3. Approximation formula error.

No. Of Pages	Avg. Latency Times	Approximation formula Latency Times	Error Value
1	1	1	0.00%
2	1.5	1.5	0.00%
3	2.3333	2.3333	0.00%
4	3.25	3.25	0.00%
5	4.2	4.225	2.50%
6	5.1667	5.2	3.33%
7	6.1429	6.175	3.21%
8	7.125	7.15	2.50%
9	8.1111	8.125	1.39%
10	9.1	9.1	0.00%

From Table 3, we know that the difference between the approximation formula value and the real value is small. The average initial latency time is 43.225, while the file has about 45 pages.

Hence, we propose the data allocation method to improve the data presented by a proper sequence order. Moreover, it can extend the data storage characteristics of NandFlash. Then, we improve Figure 4 structure into the structure of Figure 7, in which our proposed internal packing scheme algorithm is briefly denoted by $IPS_{\{our\}}$. The reading time of $IPS_{\{our\}}$ is 12us+ (A1'+A2'+A3'+A4'+A5')*50ns, using the same example in Figure 4; the reading time of $IPS_{\{our\}}$ will reduce about 48us compared with $IPS_{\{Best\text{-}Fit\}}$. Moreover, $IPS_{\{our\}}$ performance exhibits more in the file structure with large size, such as the applications of DSC, USB flash device, or MP3.

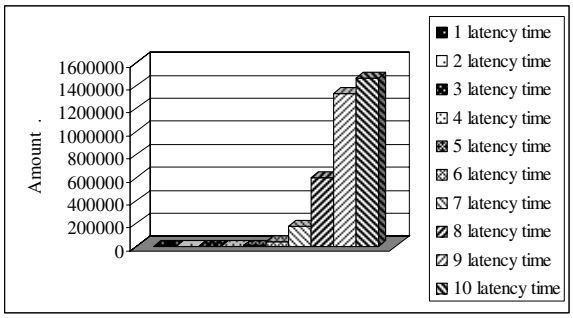

Figure 5. The distribute diagram of a file with 10 pages.

Figure 8, in which a block can store 45 compressed pages, explains the reading time between $IPS_{\{Best\text{-}Fit\}}$ and $IPS_{\{our\}}$, while we also use the average compression 140.8% ratio of $IPS_{\{Best\text{-}Fit\}}$. Here the reading time of $IPS_{\{Best\text{-}Fit\}}$ is about 1.6 times, which takes an extra 4.05ms, to $IPS_{\{our\}}$ algorithm while the file size is 180KB. Then, the reading time of $IPS_{\{Best\text{-}Fit\}}$ is 60.8ms, with the file size of 1MB; this is also about 1.6 times of $IPS_{\{our\}}$, which represents an extra 23ms cost. The above reading cost is

just the initial latency time, which does not contain the decompression time and other costs. Therefore, one of the important design factors is to consider the performance of the reading bandwidth, which becomes an important issue in design.

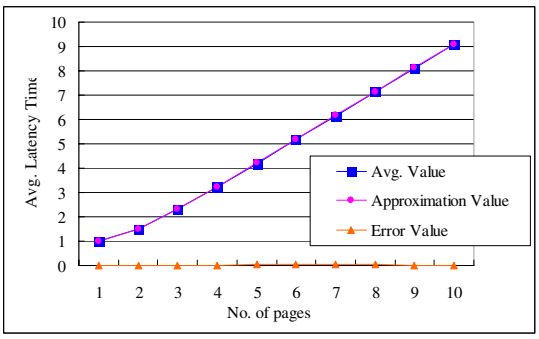

Figure 6. Different file size with the different initial latency times and approximation curve in addition to its difference value.

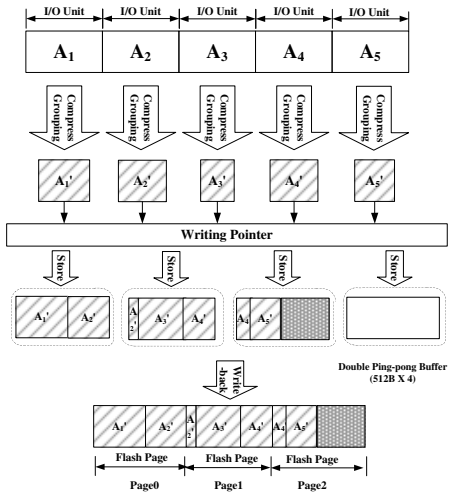

Figure 7. $IPS_{\{our\}}$ structure.

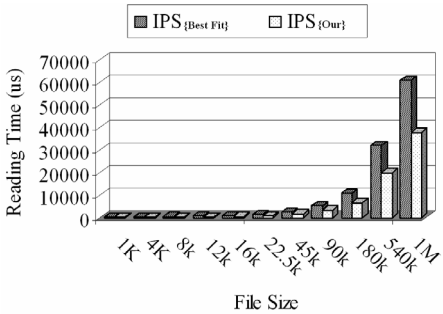

Figure 8.Comparison of the reading file time between $IPS_{\{Best\text{-}Fit\}}$ and $IPS_{\{our\}}$.

$IPS_{\{our\}}$ configuration can overcome the $IPS_{\{Best\text{-}Fit\}}$ disadvantage. We just modify that the recoded method of data chain is changed into the page address in addition to its offset to represent the starting address. Such advantage can reduce internal fragmentation, promote space utilization, and use smaller buffer to reduce its cost.

The comparison table between $IPS_{\{Best\text{-}Fit\}}$ and $IPS_{\{our\}}$ is shown in Table 4. The memory of Best-Fit requires more space to save its compressed data and then compare in addition to finding the optimal configuration to operation [9]. Since our proposed structure involves a continuous configuration, we just need a Ping-Pong buffer, which can also reach the pipeline performance, such that it can reduce the large memory requirement. Then, the bandwidth using the continuous configuration, which is the continuously reading property, is wider than those in the literature [9]. There are some internal fragmentations using the Best-Fit method, whose fragmentation degree of a page unit is shown as in Table 4. Our proposed model can reduce the degree of the internal fragmentation, which happens when it turns off the devices such that it will force the buffer data to update NandFlash. In this case, it will generate less than 512B internal fragmentations and immediately improve the compression ratio.

Table 4. Comparison table between $IPS_{\{Best\text{-}Fit\}}$ and $IPS_{\{our\}}$

	$IPS_{\{Best\text{-}Fit\}}$	$IPS_{\{our\}}$
Buffer size (RAM)	512B × 320	512B × 4
Property	Scatter	Continuous
Reading speed	Slower	Faster
Internal fragmentation	$\sum_{i=1}^{n} Fragmentation_i$	Smaller than 512B

The relation specification between NandFlash and SmartMedia cards is shown in Table 1. The specifications of SmardMedia cards are downward compatibility. Therefore, $IPS_{\{our\}}$ can be directly applied to 128MB, 64MB, 32MB, 16MB, or 8MB of SmartMedia cards. Moreover, the structure of NandFlash is the same as that of SmartMedia cards [12]-[14]. Therefore, $IPS_{\{our\}}$ can be applied to a general NandFlash.

5. Conclusions

We propose a new $IPS_{\{our\}}$ structure in this paper, which not only reduces the buffer space but also improves the compression ratio over previous methods in the literature [9]. Our model uses the continuous configuration method; the data in the buffers using the page I/O is written to NandFlash. In this method, each memory can be fully used and the reading speed can be improved greatly. The average reading time of $IPS_{\{our\}}$ is about 0.625 times of $IPS_{\{Best\text{-}Fit\}}$, if the file size is greater

than 8K. In other words, it could not reduce the reading speed for discontinuous storage operations, when a compression method is applied to MP3 or DSC devices. Moreover, our method is more practical and meaningful since its reading speed will not be degenerated when it reads a large amount of compressed motion pictures.

Reference

[1] S. Bunton and G. Borriello, "Practical Dictionary Management for Hardware Data Compression," *Communications of the ACM*, Vol. 35, No. 1, pp. 95-104, 1992.

[2] C. C. Cheng, W. T. Huang, C. T. Chen, C. H. Chen, "The Design and Implementation of Flash File System Management," *Journal of NTUT*, Vol. 36-2, pp. 43-62, March, 2003.

[3] F. Douglis, R. C'eres, F. Kaashoek, K. Li, B. Marsh, and J.A. Tauber, "Storage alternatives for mobile computers," *Proc. OSDI94*, 1994, pp.25–37.

[4] W. T. Huang, C. T. Chen, C. C. Cheng, C. C. Liu, "The Nand Type Flash Memory Management Model based on the Block Base," *Journal of NTUT*, Vol. 37-1, Accepted, March, 2004.

[5] W. T. Huang, C. T. Chen, C. C. Cheng, J. C. Kuo, "To Study and Implement the Faster and Lower Cost Flash Memory," *Journal of NTUT*, Vol. 37-1, pp. 77-89, March, 2004.

[6] W. T. Huang, Y. S. Chen, C. T. Chen, C. C. Cheng, "The Flash Memory Management Model of the Higher Reliability and Lower Cost-Benefit," *Journal of NTUT*, Vol. 37-1, pp. 61-75, March, 2004.

[7] H. J. Kim and S. G. Lee, "An Effective Flash Memory Manager for Reliable Flash Memory Space Management," *IEICE Trans. Information & System*, Vol. E85-D, No. 6, June 2002, pp. 951-964.

[8] M. Kjelso, M. Gooch, and S. Jones, "Design and Performance of a Main Memory Hardware Data Compressor," *In Proceedings of the 22nd Euromicro Conference, IEEE Computer Society Press*, pp. 422-430, 1996.

[9] K. S. Yim, H. Bahn, and K. Koh, "A Flash Compression Layer for SmartMedia card Systems," *IEEE Trans. on Consumer Electronics*, Vol. 50, No. 1, Feb. 2004.

[10] K. S. Yim, J. Kim, and K. Koh, "Performance Analysis of On-Chip Cache and Main Memory Compression Systems for High-End Parallel Computers," *Proc. of International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'04)*, Las Vegas, USA, June 2004. http://cares.snu.ac.kr/Download/yim_pdpta_survey.pdf

[11] Samsung Corp., Flash SmartMedia File System, Memory Databook, 2000.

[12] Samsung Corp., "K9F1208U0M-YCB0,K9F1208U0M-YIB0, K9F5608U0M-YCB0 Flash Memory," Data Sheet, 2001. http://www.samsung.com

[13] Samsung Corp., "K9W8G08U1M Flash Memory," Data Sheet, 2003. http://www.samsung.com

[14] Toshiba Corp., "TC58256AFT 256-MBIT (32M* 8 BITS) CMOS NAND E2PROM," Data Sheet, 2001.

[15] R. D. Yates, D. J. Goodman, R. D. Yates, D. J. Goodman, "Probability and Stochastic Processes: A Friendly Introduction for Electrical and Computer Engineers," 30 July 1998.