Multimed Tools Appl (2015) 74:6287–6302
DOI 10.1007/s11042-014-2122-z

NAND flash memory system based on the Harvard buffer architecture for multimedia applications

Cheong Ghil Kim · Kuinam J. Kim · JungHoon Lee

Received: 14 March 2014 / Accepted: 20 May 2014 / Published online: 12 June 2014

Abstract The main purpose of this research is to design a new memory architecture for NAND flash memory to provide XIP (execute in place) for code execution as well as overcome the biggest bottleneck for data execution. NOR flash for multimedia application is particularly well suited for code storage and execute-in-place (XIP) applications, which requires high-speed random access. While NAND flash provides high density and low cost data storage, it is not applicable to XIP applications due to the page access and long access latency. To overcome these limitations, NAND flash can be exploited as code memory for XIP by using SDRAM/SRAM buffer. In order to design the code memory, we proposed a NAND flash with a dual instruction buffer. Furthermore, another enhancement was proposed for the overall system performance by applying a data buffer system to the existing NAND flash memory to reduce the number of write and erase operations; otherwise which could be the biggest bottleneck in a flash memory system. In conclusion, the proposed NAND flash buffer system based on Harvard architecture is operated as main memory as well as the lowest storage device for mobile multimedia system. According to our simulation results, write and erase operations are approximately 60 % and 68 % less than other unified buffer systems, respectively, with two times more space. In addition, the average memory access time is improved by approximately 75 % compared with other unified buffer systems.

Keywords Flash memory · XIP (execute in place) · Buffer system · Harvard vs von neumann architecture · Multimedia applications

C. G. Kim
Department of Computer Science, Namseoul University, 91 Daehak-ro, Seongwhaneub, Seobuk-gu, Cheonan, Chungnam, South Korea
e-mail: cgkim@nsu.ac.kr

K. J. Kim
Department of Convergence Security, Kyonggi University, Iui-dong, Yeongtong-gu, Suwon-si, Gyeonggi-do 443-760, South Korea
e-mail: harap123@hanmail.net

J. Lee (✉)
ERI, Control and Instrument Engineering, GyeongSang National University, 501 Jinju-Daero, Jinju, Gyeongnam 660-701, South Korea
e-mail: leejh@gsnu.ac.kr

1 Introduction

Nowadays, nonvolatile memory is being widely used in personal electronic devices such as smart phones, MP3 players, and tablet PCs because of its low price and high degree of integration along with its strong resistance to shock [12]. Nonvolatile memory can be divided into two categories: NAND- and NOR-type flash memory. They are similar in their cell structure for byte storage, but there are pros and cons between them due to their difference in how their cell assembly is composed of [5].

The most distinguishing feature between NAND and NOR flash memory is the initial access time. The initial access time of NAND is 25 μs while that of NOR, depending on its reading mode, is approximately 50~100 ns. This is approximately a 250- to 500-fold difference between the two [15, 19]. After the initial access, both NAND and NOR flash take a similar amount of time for sequential data reading. This means that NAND flash has a great disadvantage in terms of indirect access time. The features of NAND and NOR are summarized in Table 1.

Due to these features, NOR flash memory is mainly used to store instruction codes for operation, while NAND is used for data storage. However, NAND does have more economical benefits. It is approximately 30~40 % cheaper than NOR flash memory [18]. Therefore, there is ongoing research into enhancing the access time of NAND flash using SRAM/SDRAM [4, 11]. Another study is attempting to maximize NAND flash memory performance by utilizing the intelligent buffer system in the flash memory [9]. In particular, reducing the writing frequency of flash cells can effectively decrease the number of erase modes, which is the largest overhead in flash memory, thereby overcoming the disadvantage of flash cell use limitations. Although there has been a lot of research on enhancing NAND flash performance by using various buffer systems, no studies applied the cache memory effects on the Harvard or von Neumann architecture [2] to flash memory systems.

The proposed NAND flash system is composed of two separate buffers to exploit the characteristics inherent in each module. The proposed instruction and data buffer in the NAND flash memory are well suited for code and data execution, respectively Furthermore, instructions and data have different localities. While instructions sequentially access data due to its spatial locality, data have both spatial and temporal localities. In cache memory, two locality features display drastically different characteristics depending on block size. The spatial locality features can be exploited by increasing the block size and bringing in a large quantity of data at once in a single fetch. However, this may reduce the number of entries in a cache. On the other hand, the temporal locality is the principle that memory locations in a program are likely to be accessed again in the near future. Its performance can be optimized by increasing the number of cache entries at the compromise of decreasing the block size. However, a unified cache system cannot satisfy both locality characteristics at the same time [14, 17].

This paper presents a new high-performance NAND flash memory system based on the Harvard architecture. The Harvard architecture uses physically separate memories for their

Table 1 Pros and cons of NAND and NOR flash memory

	NAND flash memory	NOR flash memory
ProsCons	Fast read/write performance Convenient mass storage	Fast random access time Random write in byte
	Slow indirect access time Inability of write in bytes	Slow write/erase performance Relatively difficult mass storage

instructions and data. In the case of flash memory with a buffer system, separate data and instruction buffers can lead to diversification of the flash architecture and achieve high performance by effectively overcoming the limitations of a unified buffer. Especially, with XIP functionality in NAND flash, the cost of memory system can be reduced since the NAND flash can be used not only as data storage but also as code storage for execution. Therefore, we can obtain cost-efficient memory systems with high performance and low power consumption.

2 Related work

Most of the existing research on flash memory has focused on cell research, write/erase reduction, and the mapping algorithm for cell data, etc. Consequently, there has been no research regarding NAND flash memory combined with a dual buffer system based on the Harvard architecture.

Among the studies on utilizing the existing buffer system, the smart buffer system [8] and Huang [3] enhanced the performance of the overall system by applying a buffer system to the existing flash memory to reduce the number of write operations, which is the biggest bottleneck in a flash memory system. A smart buffer consists of a buffer with a large page size fetched from the flash memory and a buffer that stores data discarded from the memory. A smart buffer reduces the write operation frequency of the flash memory and guarantees a fast memory access time. On the other hand, Huang utilized one writing buffer and two reading buffers to decrease the write operation frequency and improve the system performance. Data requests from the write operation are stored in the write buffer and read operation requests are stored in the read buffer.

Recently, researchers have proposed many buffer replacement policies. The flash aware buffer policy (FAB) [6] maintains a block level LRU (least recently used) list, in which pages with the same erasable block are grouped together. When a hit occurs on a page, the group containing the page is moved to the beginning of the LRU list. When a miss occurs, the group that has the largest number of pages will be selected as victim and all dirty pages in this group will be paged out. FAB is mainly used in the applications of portable media players where most write requests are sequential. Being different from other buffer policies FAB uses an internal RAM of SSD as a buffer to change random write to sequential write to improve the write efficiency and reduce the number of erase operations. These works just focused on data.

In industry (http://www.m-sys.com/files/documentation/doc/Mobile_DOC_G3_DS_Rev1.0.pdf), NAND XIP is implemented using a small buffer size and I/O interface conversion, but the XIP area is limited to boot code. The OS and application codes should be copied to the system memory at booting time. Park [11] created a low-cost flash memory package for managing instructions by replacing the NOR flash memory with a NAND flash memory and SRAM based on XIP (Execute In Place). This architecture consists of DRAM and a buffer with a victim cache structure. Either the victim buffer or DRAM is chosen for storage based on the reference pattern of the flash memory cell. That is, pages with many references are stored in the victim buffer and pages with fewer references are stored in DRAM. This architecture achieved not only fast memory access but also cost and energy efficiency in the flash memory by storing frequently referred pages in the buffer.

The above-mentioned research on data in flash memory such as the smart buffer, Huang, and FAB successfully enhanced the system performance by overcoming the biggest bottleneck problem of write and erase operations on flash memory cells via the use of a buffer system or replacement policy. On the other hand, Park's research focused only on instructions in flash memory. As mentioned above, although much has been achieved in enhancing flash memory

performance through the use of a buffer in data NAND flash, there is a great lack of research on instruction NAND flash. Furthermore, there have not been any studies carried out on separating the buffer into data and instruction type for flash memory storage and determining the most suitable buffer structure for the properties of data and instruction.

Generally, the buffer system may increase the chip area and cost. Park and Huang used a 64-KB buffer and a 4-MB read/write buffer, respectively. However, the proposed buffer system overhead turns out to be negligible because it applies to only a 5-KB data buffer and a 3-KB instructional buffer while the one-page buffer size in conventional NAND flash is generally 2 or 4 KB.

3 Flash buffer system based on the harvard architecture

3.1 Motivation and major factors for designing the flash memory system

The objective of this research is to design a high-performance flash memory architecture by applying a high-level system concept to the lowest storage device: this can be achieved by devising an instruction and data buffer for the NAND flash structure. Figure 1a illustrates the previous buffer systems for NAND flash memory such as oneNAND flash memory which can provide cost-effective storage in mobile embedded systems, e.g., mobile phone or digital camera. [7]. Here, a unified SRAM/SDRAM buffer is attached onto the NAND memory cell for instructions and data. However, as Fig. 1b shows, a dual buffer system with a buffer for instructions and another buffer for data will not only achieve more efficient use of spatial and temporal locality but also maximize buffer efficiency at low cost.

Here are three major factors to consider when designing a NAND flash memory system.

1) NAND Flash does not allow a page to be overwritten without first erasing its enclosing block. While a new request on the page update is written over a pre-existing file in a hard disk and other types of memory system, the page update in the flash memory is not flexible. Moreover, write and erase operations are slower than read operations; this limitation significantly decreases write performance.

2) Data information in NAND flash memory can be lost or damaged if the number of write operations in specific sectors exceeds a certain pre-defined maximum count. Unlike hard disks or other memory devices, flash memory has a limitation on the number of write operations for each page. In order to overcome such limitations, a technique for minimizing the number of

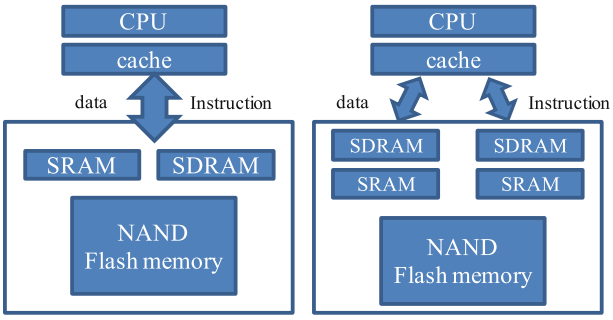

Fig. 1 NAND flash buffer system based on the Harvard architecture

write operations to the flash cell must be considered by designing a NAND flash memory system.

3) In NAND flash memory, the operating speed must be very fast to run various types of applications. In particular, fast access must be guaranteed to carry out indirect access (random access). Among the different types of flash memory, the NOR shows approximately a 100-fold faster random access speed than the NAND, while the NAND has the benefit of being approximately 30~40 % cheaper. Therefore, to construct the basic NAND flash memory architecture, a fast operation speed in terms of indirect access must be considered as one of the most important factors.

Increasing the buffer hit ratio can serve as one efficient way by applying the above-mentioned three factors when designing a new flash memory system. If data and instructions requested are hit in the buffer, then they can be read from fast SRAM/SDRAM. Thereby, the execution speed of the flash memory increases greatly.

This may be particularly efficient in reducing the write operation frequency and the largest overhead - the writing latency in flash memory - by having a buffer system which carries out the write operation instead of the flash memory cells. Consequently, a reduction in writing operations will lead to a reduction in erasing operations in the flash memory cells. Therefore, it is very crucial to enhance the buffer hit ratio and to reduce the erase operations through the use of a buffer system.

Research on buffer page size has already made considerable progress in the study of cache memory. For instance, for an instruction cache, which possesses a high spatial locality, a large page size would be ideal. However, the focus of this study is to establish the optimal page size within a fixed buffer size. Therefore, this paper designs a new flash architecture by integration with a new instruction buffer structure and mechanism. In the case of the data cache, a victim buffer with a simple architecture and mechanism can guarantee high performance in the memory system. However, the flash memory shows a better performance than the fast read operation by reducing the erase/write operations. This is due to the fact that the write and erase operations are approximately several tens of folds and thousands of folds slower, respectively, when compared to the read operations. Therefore, this paper proposes a new operational mechanism to reduce the write/erase operations for data.

3.2 Instruction flash memory

3.2.1 Efficiency of instruction buffer in flash memory

Existing typical NAND flash memory is composed of flash memory cell and a READ/WRITE register for page size. Specifically, in NAND flash memory, read and write operations execute by page unit and sequential access of such operations take 20 ns. This access time is 1,000 times faster than the indirect access time that is approximately 20us. Unlike data, instruction carries out read operation only; therefore, it can effectively improve system performance by reducing the indirect access time. Basically, instructions have a greater spatial locality due to sequential reading while the program is running.

What is special about spatial locality is that it can take advantage of a large fetching size very effectively. Existing NAND flash memory can use spatial locality effectively because it has one READ/WRITE register (e.g., 2 KB). However, although the READ/WRITE register in flash is efficient for spatial locality in sequential access instructions, its shortcoming is that it needs re-access to the flash memory cell upon branch instruction caused by a single register.

Figure 2 shows the average memory access time of flash memory when various types of instruction buffer are applied to NAND flash. The buffer size in each structure is only 4-Kbyte

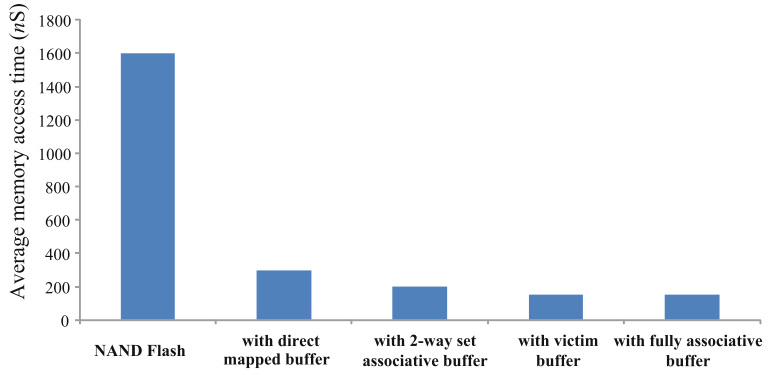

Fig. 2 Average memory access time of existing flash memory and flash memory equipped with various instruction buffer structure

SRAM and the fetching size from a READ/WRITE register to the buffer is 32 bytes, which is the minimum size in each buffer. We assumed that the NAND flash buffer system is operated as main memory as well as the lowest storage device for mobile embedded system. As shown in the figure, a 4-Kbyte direct-mapped buffer reduces the average memory access time of the existing flash memory by up to 80 %. Furthermore, flash memory equipped with a 2-way set associative buffer, victim buffer, and fully associative buffer reduced the average memory access time by about 90 %. In particular, the victim buffer and fully associative buffer are simple structures in existing cache memory which can improve system performance by reducing the conflict miss. Therefore, as seen in Fig. 2, when the structure from an existing buffer memory is applied to flash memory, there's a great improvement in system performance.

3.2.2 New instructional buffer architecture and operation algorithm

Figure 3 shows the proposed NAND flash memory with the instruction buffer system. This architecture is divided into 3 main parts: standard NAND flash memory with a 2-KB READ/

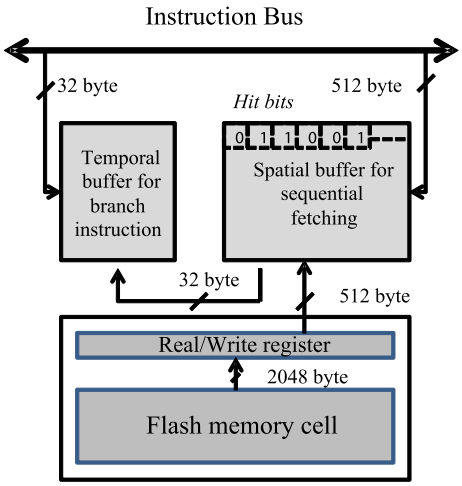

Fig. 3 Proposed instruction buffer system of NAND flash memory

WRITE register, spatial buffer, and temporal buffer. NAND flash memory is the same as the standard flash structure, and each page is set at 2 KB in accordance with the current trend of mass storage. The dual buffer architecture for instruction consists of spatial and temporal fully associative buffers.

A spatial buffer can have a large fetching size in order to effectively utilize spatial locality, the basic feature of instruction, while a temporal buffer selectively saves referenced pages. Such a temporal buffer can enhance system performance considerably especially in terms of branch instructions.

The fetching size of the spatial buffer is set to be approximately n times greater than that of the temporal buffer. Here, the fetching size of the temporal buffer is 32 bytes and the spatial buffer is made up of n*32 bytes for more efficient spatial locality. In addition, the spatial buffer has n hit bits per each entry for saving referenced instructions onto the temporal buffer selectively. Here, we consider a large fetching size of the spatial buffer to be 512 bytes. Therefore, in a fetching entry of the spatial buffer, there will be a total of 16 hit bits for the temporal buffer.

When an instruction page is demanded from the flash memory, first the dual instruction buffer is accessed. If access to the spatial buffer is a hit, a hit bit of a corresponding small page (e.g., 32 bytes) is renewed; that is, the hit bit is set as 1, and the requested 32-byte page is sent to the on-chip instruction cache. In addition, upon a temporal buffer access hit, the requested 32-byte page is directly sent to the on-chip cache without any further actions. If an access misses both buffers, the 512-byte page including the requested instruction is stored in the spatial buffer from the flash memory. Here, the hit bit of the referenced small page within the fetching size of the spatial buffer is renewed as “1”. If the spatial buffer is not full, the page is stored in the spatial buffer without removing an old page and the corresponding hit bit is set to one. However, if all the entries in the spatial buffer are valid, one entry is chosen, based on the FIFO algorithm. The referenced small pages in one replaced large page are selectively transferred to the temporal buffer; only the small pages whose hit bit is set as “1” in the spatial buffer are selectively transferred to the temporal buffer.

3.3 Data flash memory

3.3.1 Data features of NAND flash memory

In general, data carry out write operations. In NAND flash memory, data must be written in invalid pages because overwriting is impossible on a data-written page upon access. A write operation takes 200us, which is 10,000 times slower than a sequential read operation. If there are no nullified pages available in the block of NAND flash memory for the write operation, all valid pages in the block must be written in another invalid flash block and the block must undergo an erasing operation to be processed as an invalid flash block. An erase operation, which has the largest overhead in NAND flash memory, takes 1.5 ms. Therefore, being different from instructions, a data buffer of NAND flash memory must significantly reduce the number of write and erase operations in order to achieve high performance enhancement.

In this research, a data buffer is used to investigate the features of NAND flash memory. Through previous simulations, we knew that a direct-mapped buffer, 2-way associative buffer, and 4-way associative buffer display very low performance at small buffer size. This is a very different characteristic of a data buffer compared with an instruction buffer. For this reason, 8-way, victim, and fully associative structures are selected and considered. As a result, a 16-KB full associative data buffer is selected for its good performance at low capacity.

NAND flash memory with a data buffer has different features of spatial and temporal locality depending on the characteristics of the program. Spatial locality is advantageous for sequential access; therefore, it performs effectively with a large fetch size as the chance of referencing the related data of the recently referenced data increases. Temporal locality means there is a high chance of re-referencing the previously referenced data in a short time. Therefore, a unified fully associative buffer cannot use temporal and spatial locality effectively. Furthermore, using a buffer with a small fetch size in NAND flash memory could increase the temporal locality, but will result in increased writing operations in the flash memory.

3.3.2 New data buffer architecture and operation algorithm

In order to be consistent in the architectural design, a similar buffer system is used in both the instruction and data buffer. At the same time, a mechanism for reducing the write and erase operations is provided in order to maximize the performance enhancement.

As Fig. 4 illustrates, the basic structure of the data buffer is a dual buffer architecture that includes a spatial buffer with a large fetching size and a temporal buffer with a small fetching size. Spatial buffer fetching size is the sum of n small page sizes. For instance, when the temporal buffer page size is 64 bytes, the spatial buffer page size could be n * 64 bytes. Therefore, per spatial buffer, there are n hit bits and n dirty bits. If the spatial buffer page size is 512 bytes and the temporal buffer page size is 64 bytes, each large page of the spatial buffer may have 8 hit bits and 8 dirty bits per each entry.

A temporal buffer with a small fetching size can request a large quantity of writing operations in a single page, not like instructions. If a small page (e.g., 64 bytes) is only written into the flash memory, the corresponding page in flash memory is loaded into a read/write register in advance, and the modified small page is updated into the corresponding page. And then the write operation can be completely performed because a page is the basic unit of write operations. Therefore, in order to reduce the many writing operations, we suggested a new method in which all written small pages included in a 2-KB page boundary are searched in the temporal buffer and written into the flash cell simultaneously. The operational model for the proposed data buffer management is described in detail as follows.

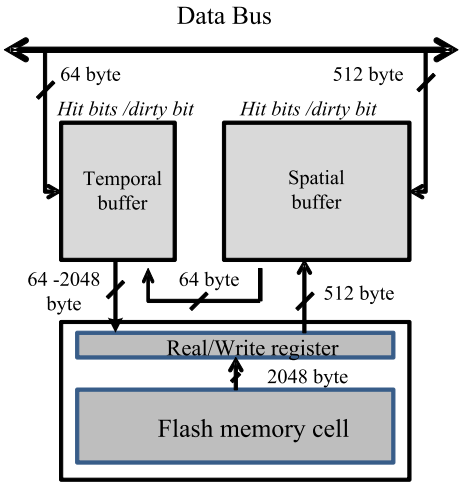

Fig. 4 Proposed data buffer system of the NAND flash memory

Read operations are processed in the same way as the instruction buffer. If a read access to the spatial buffer is a hit, the requested 32-byte page is transmitted to the cache memory and the corresponding hit bit is set to mark it as referenced. The temporal buffer has a similar operation algorithm to the instruction temporal buffer except that it has only dirty bits.

In the case of writing, if a write access to the spatial buffer is a hit, a write operation is performed and the dirty bit and the hit bit of the corresponding small page are set. If the spatial buffer is full, the oldest entry is replaced according to the FIFO policy. Here, referenced small pages from the extracted large page are selectively stored in the temporal buffer. A write operation into the flash cell only occurs in the modified page extracted from the temporal buffer. If there are no more invalid entries in the temporal buffer and data are renewed in the extracted page of the temporal buffer, all the small pages that belong in the same basic page are searched and stored in flash memory at the same time. Since a READ/WRITE register for a basic page in flash memory is 2 KB, it is possible to collect all modified small pages on the same page. The temporal buffer has only 32 entries of 64 bytes because of its 2-KB size. Therefore, in order to search all written small pages that belong in the 2-KB page boundary, 31 times the search operations by the temporal buffer's CAM tags take a very short time compared with the writing time. This single write operation can reduce the frequency of writing operations effectively. Furthermore, if the erase operation occurs in NAND flash, the dirty pages in the temporal buffer including the erase block are all updated in flash memory cells and the dirty bits of the selected page are set as 0.

If a miss occurs in both buffers, a large page including the missed small page is brought into the spatial buffer from the flash memory. However, any small pages in the temporal buffer may be included in this large page. Such page storage wastes the temporal buffer space. Furthermore, if these pages in the temporal buffer are already modified, such pages can disturb the memory coherence. Therefore, the proposed buffer system searches the corresponding small pages in the temporal buffer, and the small pages are invalidated at the same time. If any small page in the temporal buffer is a dirty page, the page is updated at the same place in the large page of the spatial buffer. This operation can use the temporal buffer space efficiently. In addition, it can also guarantee data coherence and reduce write and erase operations. We consider two cases, depending on whether the spatial buffer is full or not.

Figure 5a shows the existing state of each buffer. As shown in Fig. 5b, if any data in the small page A (4) are accessed and failed in both buffers, the large page A (1,2,3,4) is fetched into the spatial buffer. At the same time, the small page A (5), which was referenced among the removed large page A (5,6,7,8) in the spatial buffer, is stored in the temporal buffer.

If at least one entry in the temporal buffer is in the invalid state, the replaced small page A (5) is stored into the temporal buffer without replacing any small page in the temporal buffer. As our example, if the temporal buffer is full, D (1) is extracted from the temporal buffer according to a FIFO policy. As shown in Fig. 5c and d, the controller searches the tags of the temporal buffer to detect whether any of the small pages belonging to the fetched large page A (1,2,3,4) are already present in the temporal buffer. Therefore, the small pages A (1) and A (3) in the temporal buffer are invalidated and they are updated in the spatial buffer. This mechanism can make better use of a temporal buffer space. Finally, a write-back operation must be performed in the flash cell since D (1) is already modified as shown in Fig. 5b. D (5) belongs to the basic page D and undergoes the write operation on flash cells along with D (1). Finally, the D (5) dirty bit is set as 0. However, the non-modified D (3) does not take part in the main operation. Figure. 5e shows the state of the temporal and spatial buffer after the fetching and writing operation of NAND flash memory.

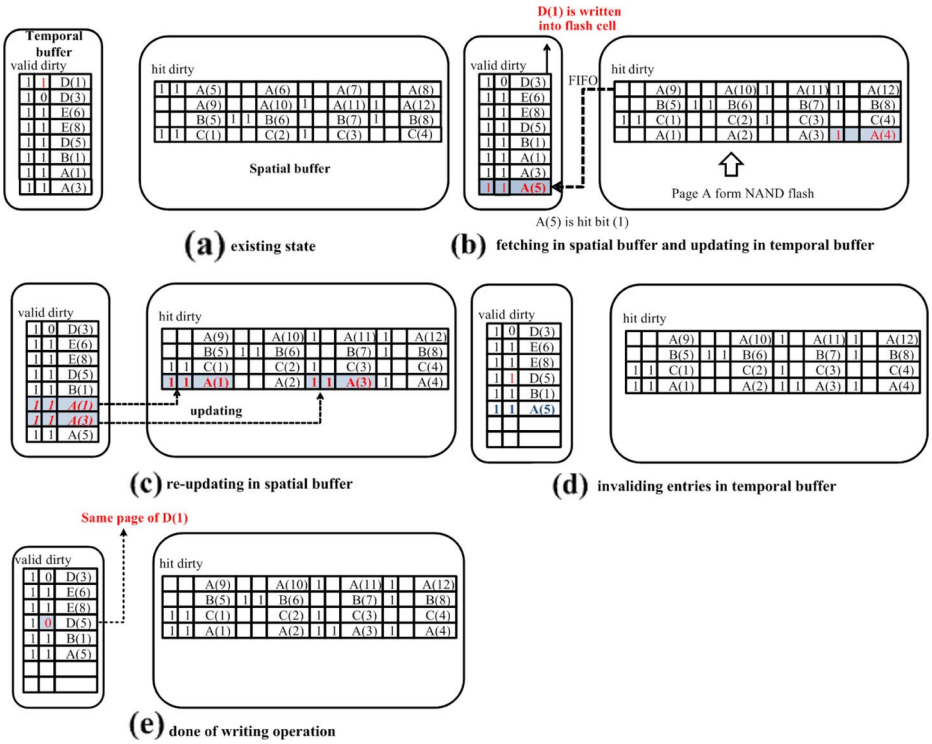

Fig. 5 Miss handling with fetching and writing operation

3.4 Harvard architecture-based NAND flash memory

The purpose of this research is to design a high-performance NAND flash memory system by using a small instruction buffer and a small data buffer. All of the commercialized buffers in use have a single unified buffer, such as the Von Neumann architecture, in which both instructions and data are stored in one buffer. On the other hand, the Harvard architecture-based system is designed to be suitable for features of instruction and data buffers to achieve performance improvement effectively. In this research, we suggest a new Harvard architecture-based NAND flash memory buffer system. The structure of both instruction and data buffers is identical, as shown in Fig. 6. The operating mechanism and structure of these buffers were explained in the previous sections.

4 Performance evaluations

This section describes the simulation conditions and results in detail. Computer architects and performance analysts are well aware of the workload drift phenomenon, and to address it, they typically collect benchmarks to represent emerging workloads. Examples of recently introduced benchmark suites covering emerging workloads are Media Bench [10] for multimedia workloads, MiBench [1] and EEMBC (Embedded Microprocessor Benchmark Consortium) [13] for embedded workloads. Therefore, the media benchmark is used to evaluate the system performance for multimedia devices.

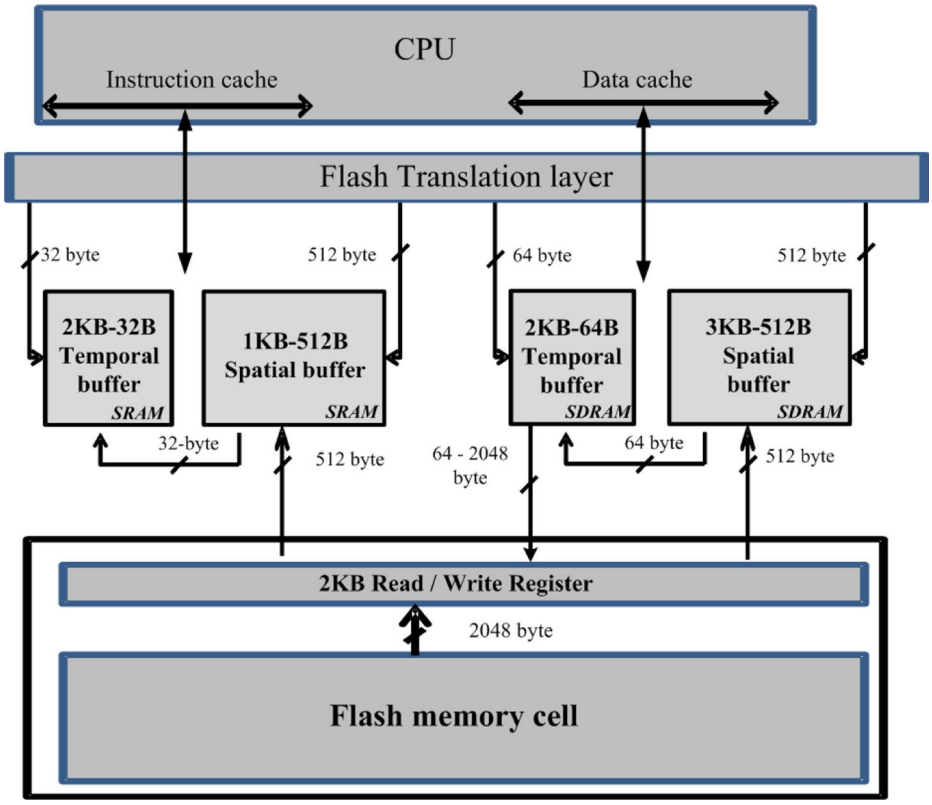

Fig. 6 Harvard architecture-based NAND flash memory buffer system

For the simulations, we devised a prototype board and used a simplescalar3.0 [16] cache simulator with pattern analysis. The prototyping board consists of an ARM9-based microprocessor, SRAM for an instruction buffer, SDRAM for a data buffer, and NAND flash. It is not only used to implement a real buffer configuration on FPGA

Table 2 Simulation parameters

System parameters	Value
CPU clock	200 MHz
L1 I/D cache	4 way, 32-byte line, 8-KB
Bus width	16 bits
Page size	2 Kbytes
Block size	128 Kbytes (64 pages)
Erase operation time	1.5*ms*
Write operation time	200*us*
Indirect read time	20*us*
Serial read time	20*ns*
SRAM Buffer access time	50*ns*
SDRAM Buffer access time	150*ns*

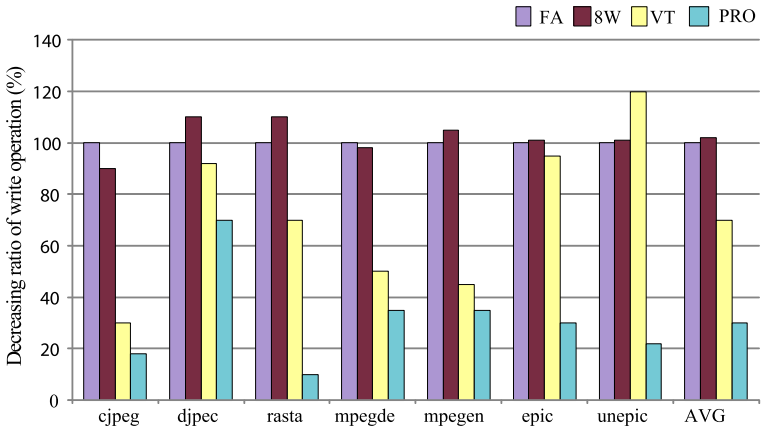

Fig. 7 The percent of write operations eliminated by each buffer type

but also to gather a flash memory address trace from running applications for the buffer simulator. The detailed specifications of the main processor and NAND flash memory are shown in Table 2. In addition, we used the existing ROMFS file systems and hybrid mapping algorithm.

The proposed flash memory page size is 2 KB and the block size is 128 KB, resulting in a total of 64 pages. Using the buffer access miss ratio and average memory access time (AMAT), we analyzed and compared the proposed buffer system to other buffer systems with NAND flash memory.

After simulations with multiple different combinations of instruction and data buffer sizes, we decided on an 8-KB dual buffer with a 5-KB data buffer and 3-KB instruction buffer as the optimal small buffer size on the Harvard architecture-based buffer system. According to the simulation results, the data buffer showed the best performance when its 3-KB spatial buffer had 512-byte large pages and its 2-KB

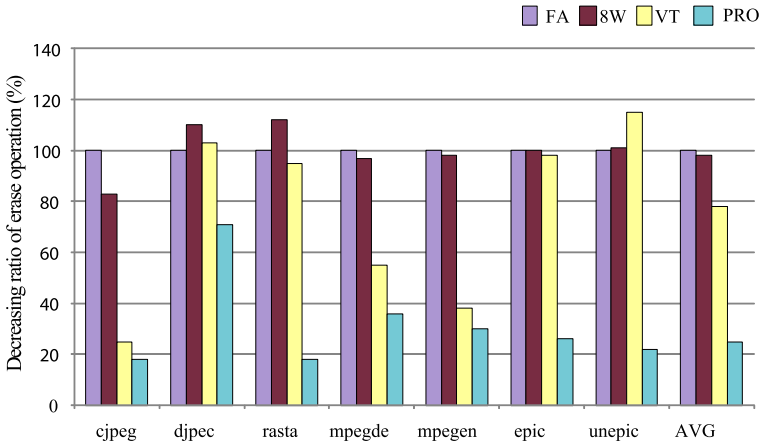

Fig. 8 The percent of erase operations eliminated by each buffer type

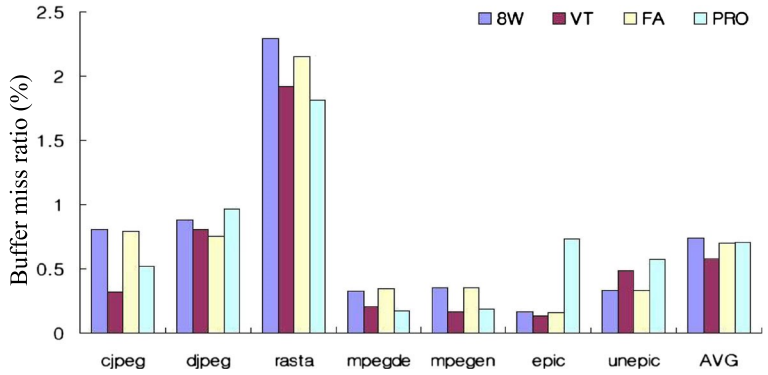

Fig. 9 Miss ratio of proposed Harvard architecture-based buffer system and Von Neumann buffer architecture

temporal buffer had 64-byte small pages; furthermore, for the instruction buffer, the best performance improvement was found at a 1-KB spatial buffer with 512-byte large pages and a 2-KB branch buffer with 32-byte small pages. Therefore, we selected a 5-KB data buffer with a spatial buffer of 512 bytes – 3 KB and a temporal buffer of 64 bytes – 2 KB plus a 3-KB instructional buffer with a spatial buffer of 512 bytes – 1 KB and a branch buffer of 32 bytes – 2 KB for the proposed buffer system.

We selected an 8-way associative buffer (8 W), victim buffer (VT), and fully associative buffer (FA) for comparison of performance. The selected buffers guarantee better performance in the data buffer because they have a larger data overhead than the instruction buffer. For the selected buffers, the most effective buffer size and fetching size are selected based on previous simulations carried out on a buffer size ranging from 8 KB to 256 KB and a fetching size ranging from 32 to 2,048 bytes. Although most structures with a buffer size of 16 KB and a fetching size of 512 bytes performed adequately in terms of miss ratio, 1,024 bytes was selected as the optimal fetching size as a result of having the best average memory access time. Only the victim buffer is 20 KB because its fully associative buffer has four entries. Figures. 7

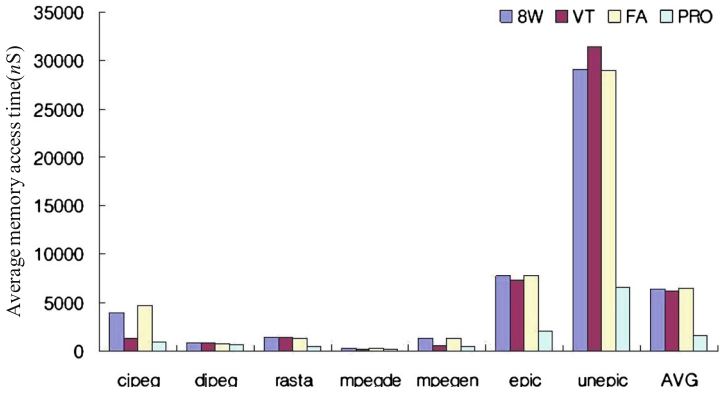

Fig. 10 Average memory access time of proposed Harvard architecture-based buffer system and Von Neumann buffer architecture

and 8 show the percentage of write and erase operations eliminated by each buffer in the NAND flash memory. The percent is normalized to the value of the fully associative buffer. The proposed buffer shows the best performance improvement in terms of write and erase operations despite its small capacity. In summary, the proposed buffer system decreases the write operations by approximately 73 %, 74 %, and 60 %, and the erase operations are improved by approximately 74 %, 74 %, and 70 %, respectively, compared with the fully associative buffer, the 8-way associative buffer, and the victim buffer with two times more space.

Figure 9 illustrates the buffer access miss ratio. Our results show that the system enables the buffer size to be reduced by a factor of two relative to other buffer structures while maintaining a similar performance. This is because the proposed buffer is able to avoid the conflict access misses upon access by both instructions and data. Figure 10 shows the most important index of performance simulation, average memory access time of the NAND flash memory. The results show that the proposed system can achieve improvements of approximately 75 % on average in the memory access latency.

5 Conclusion

In this paper, our main objective is to design a high-performance NAND flash memory architecture based on a high-level system concept. To obtain this goal, we presented a new memory architecture for NAND flash memory which can be used not only as data storage but also as code storage for execution. The proposed NAND flash memory is equipped with an instruction and a data NAND flash buffer for instruction and data characteristics, respectively, rather than having a unified NAND flash buffer of lowest system-level for storage mediums. The proposed instruction buffer system in a NAND flash memory consists of two parts, i.e., a fully associative temporal buffer for branch instructions and a fully associative spatial buffer for spatial locality. Furthermore, the proposed data buffer system is an extended combination of a fully associative temporal buffer and a fully associative spatial buffer. In order to maintain consistency in the architectural design, a similar structure and operating algorithm are used in both the instruction and data buffer. In addition, for the data buffer, a new mechanism is used to reduce the write and erase operations.

According to our simulation results, the write and erase operations are approximately 60 % and 68 % less than other buffer systems, respectively, with two times more space. The average memory access time is also improved by approximately 75 % compared with other buffer systems.

Acknowledgments Funding for this paper was provided by Namseoul University.

References

1. Guthaus MR et al. (2001) MiBench: A free, commercially representative embedded benchmark suite. Proc. of the 4th Ann. IEEE Int'l Workshop Workload Characterization, pp. 3–14
2. Hennessy JL, Patterson DA (2006) Computer architecture: a quantitative approach (4/E). Morgan Kaufmann
3. Huang W, Chen C, Chen C, Chen C (2008) Energy-Efficient buffer architecture of flash memory. Proc. of the Multimedia and Ubiquitous Engineering, pp. 543–546

4. Hyun S, Lee S, Ahn S, Koh K (2008) Improving the demand paging performance with NAND-type Flash Memory. Proc. of the International Conference on Computational Sciences and its Applications, pp. 157–163
5. Igor S (2011) Flash memories. Intech, ISBN 978-953-307-272-2
6. Jo HS, Kang JU, Pack SY (2006) FAB: flash-aware buffer management policy for portable media players. IEEE Trans Consum Electron 52(2):485–493
7. Joo Y, Choi Y, Park C, Chung SW, Chung E, Chang N (2006) Demand paging for OneNAND Flash eXecute-in-place. Proc. of the 4th International Conference on Hardware/Software Codesign and System Synthesis, pp. 229–234
8. Jung B, Lee J (2009) Flash memory system with spatial smart buffer for the substitution of a hard-disk. J of the Korea Soc of Comput and Inform 14(3):41–49
9. Lee J, Park G, Kim S (2005) A new nand-type flash memory package with smart buffer system for spatial and temporal localities. J of Syst Archit 51(2):111–123
10. Mediabench, Available from: <http://euler.slu.edu/~fritts/mediabench>
11. Park C, Seo J, Bae S, Kim H, Kim S, Kim B (2003) A low-cost memory architecture with NAND XIP for mobile embedded systems. Proc of the 1st CODES-ISSS:138–143
12. Park J, Park S, Weems CC, Kim S (2011) A hybrid flash translation layer for SLC-MLC flash memory based multibank solid state disk. Microprocess Microsyst 35(1):48–59
13. Poovey JA (2009) A benchmark characterization of the EEMBC benchmark suite. IEEE Micro 29(5):18–29
14. Przybylski S (1990) The performance impact of block sizes and fetch strategies. Proc. of the 17th Annual International Symposium on Computer Architecture, pp. 160–169
15. Samsung Elec. (2010) NAND flash memory & smart media data book
17. Sanchez FJ, Gonzalez A, Valeo M (1997) Static locality analysis for cache management. Proc. international conference on parallel architectures and compilation techniques, pp. 261– 271
18. Santarini M (2005) NAND versus NOR., available from: < http://www.edn.com/contents/images/6262540.pdf>
16. SimpleScalar LLC, SimpleScalar 3.0, Available from: <http://www.simplescalar.com>
19. Toshiba (2009) NAND vs. NOR flash memory technology overview, available from: < http://umcs.maine.edu/~cmeadow/courses/cos335/Toshiba%20NAND_vs_NOR_Flash_Memory_Technology_Overviewt.pdf>

Cheong Ghil Kim received the B.S. in Computer Science from University of Redlands, CA, U.S.A. in 1987. He received the M.S. and Ph. D. degree in Computer Science from Yonsei University, Korea, in 2003 and 2006, respectively. Currently, he is a professor at the Department of Computer Science, Namseoul University, Korea. His research areas include Multimedia Embedded Systems and AR.

Kuinam J. Kim received B.S. in Mathematics from the University of Kansas in 1988. He received M.S. in Statistics and Ph. D. in Industrial Engineering from Colorado State University 1994. He is a professor and chairman of the convergence Security Department, Kyonggi University, Korea. His research interests include Information and convergence security, cloud computing and advanced technology.

JungHoon Lee received his B.S. degree from Sung KyunKwan University, Seoul, Korea, in 1999, and the M.S. and Ph. D. degree in Computer Science from Yonsei University, Seoul, Korea, in 2001 and 2004, respectively. He is currently a professor in Control Instrumentation Engineering, GyeongSang National University, Korea. His research interests include advanced computer architectures, intelligent memory system, next flash memory, and low power technologies.