High Durability in NAND Flash Memory through Effective Page Reuse Mechanisms

Kwangyoon Lee
Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA, USA
kwl002@cs.ucsd.edu

Alex Orailoglu
Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA, USA
alex@cs.ucsd.edu

ABSTRACT

In this paper, we introduce a highly effective page reuse mechanism to reduce the amount of block erasures and page programming in NAND based primary memory architectures. The proposed techniques provide a very high rate of page reuse by effectively incorporating bit differences in page updates along with a reduction in bit unprogrammability by minimizing programming interference among adjacent pages. We also propose an effective block reclamation scheme to alleviate overall programming stress in a block so as to reduce the probability of run-time cell defects. The page reordering scheme can further increase page reusability by reducing run-time programming disturbance. The experimental results show that our proposed techniques significantly diminish the amount of block reclamation and consequently enhance the durability of the NAND flash based storage systems. Furthermore, by alleviating overall bit stress in NAND flash memory, the probability of bit failure of each cell is also significantly reduced, enabling the construction of more reliable and durable NAND flash based memory.

Categories and Subject Descriptors: C.3 [Special-Purpose and Application-Based Systems]: Real-Time and Embedded Systems

General Terms: Performance, Design, Experimentation

Keywords

NAND Flash, Primary Memory, Durability, Memory Architecture

1. INTRODUCTION

Embedded systems are serving diverse functionalities and the design of embedded systems is impacted by the consequent diversity in requirements, such as power, performance, reliability, cost, capacity and non-volatility. Memory subsystems are one of the most critical components in embedded systems design; highly customized memory system design for the dedicated systems constitutes a dominant practice in ensuring effective satisfaction of the aforementioned requirements, consequently. Memory system design can be far more simplified and effective when a single memory device can serve all the requirements.

DRAM has been a dominant memory solution for most embedded systems for several decades. However, DRAM's decade long dominance in embedded systems is significantly weakened due to its power consumption characteristics as the importance of low-power consumption is ever increasing for various types of embedded systems and especially for mobile systems for which the power budget is highly limited.

As a complete substitute of low-power primary memory for DRAM, various alternative memory technologies such as NAND, FeRAM, MRAM and PRAM [11, 7, 14, 3] have been developed and introduced but none of the alternative memories except for NAND flash memory have successfully permeated the memory market until now. There exist multiple common barriers such as cost, capacity, manufacturability and durability that have precluded the elevation of these devices to a mainstream primary memory. By successfully overcoming imposed cost, capacity, and manufacturability barriers for the last two decades, NAND flash memory has emerged as the most popular secondary memory device in embedded systems. However, the strict durability limits in NAND flash memory constitute one of the most critical factors in precluding its more aggressive adoption as a primary memory. Therefore, efficient mechanisms to extend the durability of the NAND flash memory are required to promote a NAND-based primary memory.

In this paper, we introduce an effective NAND flash page reuse mechanism to construct a durable NAND based primary memory. The durability of the proposed primary memory is significantly enhanced by boosting cell programmability and minimizing the bit screening effect. Furthermore, the probability of a cell defect due to cell aging is also reduced through the effective reduction of bits in a block. The proposed mechanism only requires minimal hardware overhead. Extensive simulation using the SimpleScalar simulator with

Table 1: ITRS 2009 Roadmap for NAND Fla Memory Technology

Device	2010	2014	2018
NAND Technology (nm)	32	22	14
Write/Erase Cycles	1E+05	1E+04	1E+04
Data Retention (years)	10-20	10	5-10
Max. Bits/Cell	3	4	4

Spec2000 benchmark suites illustrates the significant reduction in page replacements and the associated extension in lifespan of the NAND flash memory.

The rest of the paper is organized as follows. Section 2 briefly presents NAND flash based memory systems. Section 3 introduces background and motivational ideas. Section 4 delivers an in-depth introduction of the proposed page reuse mechanism including the page difference update, the bit programming mask and the page reordering scheme. Section 5 provides experimental results based on simulation and analysis. In section 6, an overall summary of the proposed mechanism is offered.

2. RELATED WORK

Over a decade, a significant amount of research has been undertaken aimed at the effective use of NAND flash memory in various storage architectures. In practice, the adoption of NAND flash memory has been limited to a certain extent and is typically consigned to secondary or tertiary data storage where the aforementioned limitations of NAND flash memory could be effectively tolerated. However, a significant amount of research has been performed to overcome these intrinsic limitations of NAND flash memory [9, 10] and extend the areas of applicable memory/storage systems such as primary memory and system buffers. Moreover, many researchers and developers still envision NAND flash memory as the most plausible unified memory device in embedded systems.

Park et al. in [13] employs NAND flash memory as a primary memory. The authors add a simple direct mapped unified cache between a processor and the NAND flash memory. However, this system requires a relatively large cache and it suffers from overall performance degradation. In [1], OneNAND^{TM}, a hybrid NAND flash memory, has been introduced as a high-performance non-volatile memory for embedded systems. In [12, 6, 5], the authors develop a NAND flash memory based code storage system by incorporating the virtual memory. However, it requires a considerable amount of VM page buffers. In [9], the authors introduce a NAND flash memory based code storage system by employing application specific hotspot information to effectively relieve the extremely long access time of the NAND flash memory. [10] reduces the number of reclamations through the use of an accurate run-time block utilization ratio but fails to reduce the actual amount of page programming. In [4], the authors introduce a new bit encoding scheme enabling two in-place page programs but the critical impact of the possible bit disturbances during the page updates is not explored.

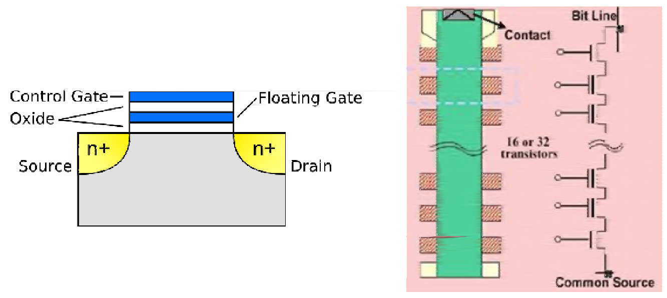

Figure 1: The NAND Cell Architecture and Bit Organization

3. BACKGROUND AND MOTIVATION

Unlike conventional memory devices whose address space is designed to be accessed randomly in byte granularity, NAND flash memory exposes pages and blocks as its basic granularity of access and it further requires both *program* and *erase* operations to write data. To represent a bit status, a floating gate of the NAND cell should be charged or discharged with respective operations named program and erase and high voltage should be applied during these operations. Consequently, the stringent limitation on the total amount of valid erasure/programming is guidelined by each NAND flash manufacturer. The technological advancement in NAND flash memory is mostly dedicated not to enhance bit characteristics or performance but to enlarge capacity by introducing a new fabrication technology or cell technology. However, based on the ITRS '09 report as can be seen in table 1, gradual degradation in durability is expected in the coming decade [2].

Most critical durability problems in NAND flash memory are imposed by cell-architectural and bit-organizational characteristics. NAND flash memory has a quite simple but effective hardware cell architecture as seen in figure 1 with the floating gate architecture engendering a highly scalable memory device. Furthermore, the cell organization is also highly effective in providing low-power and high performance. However, it displays certain levels of problems and constraints. First of all, NAND flash cells age over *program/erase* operations due to the high applied voltage needed to charge/discharge the floating gate [8]. While the integrity of up to 100,000 times is guaranteed in the current generation of SLC NAND flash memory, this constraint causes significant impact on the durability of the NAND based memory system. Secondly, there exist multiple sources of disturbances to the adjacent cells when high voltage is applied during the page programs. To minimize the impact of run-time program disturbances, page programming should be performed in sequential order within a block (generally, in ascending order of page number) and a page can be reprogrammed only up to 4 NOPs (Number of Partial Programs) in most SLC NAND flash memories. As a

Table 2: Timing Characteristics of Memory Devices

Device	SRAM	DRAM	NAND FLASH	OneNAND^{TM}
Access Unit (byte)	1	1	2K	2K
Read Bandwidth (MB/s)	2,500	1,000	17	108
Write Bandwidth (MB/s)	2,500	1,000	6.8	9.4
Guaranteed Erase Cycles	N.A.	N.A.	100,000	100,000

result, the durability of NAND flash memory is significantly exacerbated by this combination of constraints.

To enhance the durability problem in NAND flash memory, page reuse can be the most promising and effective way to reduce the overall amount of programmed pages and result in a consequent reduction of block erasures. However, in most systems, page reuse cannot be effectively incorporated due to the overly strict guidelines of the NAND manufacturers that only permit the sequential order page programming within a block, which is mandated so as to ensure no effective bit disturbance among cells in a bit-line. The already reset cells in a bit-line whose location is closer to the common source as seen in figure 1 than the cell currently being reset may impact the fidelity of the resetting operation due to the significant voltage shift imposed. This run-time effect is defined as *bit screening* in the paper and may cause dynamic bit errors during programming.

As a general page programming guideline, pages can be reprogrammed and random order page programming can be conditionally allowed provided all the three programmability criteria are met:

1. NOP constraint - the number of page programming should not exceed the NOP limitation.
2. Bit programmability - set bits can be reset but already reset bits cannot be set.
3. Bit screening - non-sequential order (random order) page program is only permitted when no bit screening effect is imposed during the programming.

In flash-based memory systems, there exist various mapping mechanisms to utilize pages and blocks in a flexible manner. Each mapping mechanism explores a distinct optimization goal and ways of re-placing pages and blocks at run-time. However, all the mapping mechanisms require a common NAND flash memory specific operation, namely, block reclamation, which is an inevitable but extremely important and complex operation in NAND flash memory to enable reuse of the blocks already programmed. Block reclamation can be initiated when no further available pages are found on the block.

In addition to the limited physical block-level durability, NAND flash memory also differs from other traditional memory components such as SRAM and DRAM in that it requires a larger granularity of access and displays highly unbalanced access characteristics between reads and writes as shown in Table 2. The physical unit of *read/write* access of NAND flash memory is 2KB and it is in general the optimal cache line size due to the extremely long write access latency in NAND flash memory. However, this can induce elevated cache replacements due to the increased conflict misses on the cache, which results in performance and durability degradation.

On-demand page requests are issued when fresh new data arrives or existing data are updated. During run-time, page update constitutes the major traffic; yet the page update frequency over the given flash devices is highly skewed as some pages experience a quite elevated rate of consecutive updates while others remain unchanged for a long period of time. Frequent page updates can easily consume all the available pages in a block and consequently trigger a high volume of block reclamations. In our previous work [10], the number of reclamations is significantly reduced by adjusting each block utilization ratio using dynamic application information. However, the total amount of page consumption remains the same.

Page reuse is the most effective technique to reduce the total amount of page consumption. However, it is extremely unlikely to reuse pages by purely updating upcoming traffic just in place. If we can incorporate though the incremental difference of the updates, there exists a high probability of page reuse provided no spatial conflict occurs in consecutive updates. We observed many page updates dealing with large regular data sets display the aforementioned behavior.

In previous research on NAND flash memory, a programmed page is considered to be in a terminal state that can be no longer utilized unless the block that contains it is reclaimed. However, these pages can be reused by removing the run-time inhibiting factors such as bit unprogrammability and bit screening by restructuring data contents, reorganizing page distribution based on its update frequency and reordering page location on reclamation. Along with previous research to minimize the reclamation amount and overhead, the page reuse technique can further enhance the usability of page resources and consequent durability of the device. Furthermore, by minimizing the number of programmed bits, the overall probability of the generation of run-time faulty cells is also significantly reduced, resulting in higher durability.

4. PROPOSED ARCHITECTURE

The page updates in the NAND flash memory layer exhibit highly non-overlapping bit patterns over iterations and the total number of bit changes in a page update of the Spec2000 benchmark programs fails to exceed 15%. Consequently, by isolating the bit changes in each page update, the total number of page writes can be significantly reduced by reusing already programmed pages or by incorporating

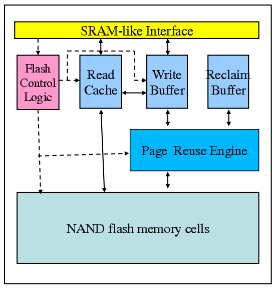

Figure 2: The Proposed Memory Architecture

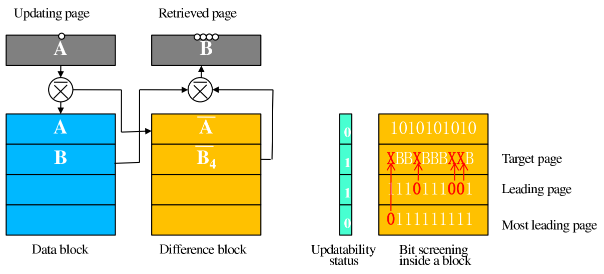

Figure 3: Bit Difference Tracking Mechanism

bit transitions in additional pages. However, the sequential order page programming constraints significantly hinder this approach by precluding random order page programming in a block. The major contributions of the proposed mechanisms lie in the introduction of the effective page reuse mechanisms. We attain this by allowing random order page programming under the fulfillment of the aforementioned bit screening constraint and by minimizing the impact of bit screening in block reclamation through effective page reordering.

The detailed architectural ideas are introduced in the subsequent subsections; the overall hardware architecture can be seen in figure 2.

4.1 Page Update Through Bit Difference Tracking

Data accesses to memory pages generally exhibit diverse access frequency and patterns. More importantly, the access frequency is highly skewed and only a small portion of pages are frequently accessed while most of the rest of the pages are rarely accessed. Furthermore, we have identified different types of bit change patterns in page updates, (1) random bit change, (2) same bit update, and (3) non-overlapping bit update. While all three types of page updates can be observed in most programs, for the frequently updated pages, the non-overlapping bit updates are the most commonly observed patterns as the big arrays in a program are in general updated partially for a given time frame. Most of these gradually changing memory arrays that are frequently evicted from cache layers constitute significant portions of the NAND flash memory write traffic.

There exist two major constraints in NAND flash memory that preclude the reuse of pages that are frequently updated: sequential order page programming and bit screening effect. However, the sequential order page programming constraint can be avoided when there is no bit screening violation as discussed in section 3. Consequently, pages can be effectively reused up to the NOP count specified by the manufacturers.

The most effective way of page reuse is to update the page in place. However, in most cases, it violates the unidirectionality constraint of bit change in page programming. Consequently, a more effective way of page update by incorporating bit differences in each page update can be extremely beneficial. As discussed, the page update that only changes a small fragment of the page with no overlapped bits is the most ideal candidate for the difference update. These page updates can be seamlessly converted into a difference update that completely satisfies the three page programmability criteria resulting in effective page reuse.

As seen in figure 3, the difference between the original page A and the updated page $\dot{A}$ is represented by $\bar{A}_1$, the result of the simple arithmetic operation, $\bar{A}_1 = A \odot \dot{A}$, where $\odot$ represents the XNOR operation. Due to the behavioral characteristics, $\bar{A}_1$ will be mostly filled with set bits except for the bits that are being modified as a result of the update. For the subsequent page $\ddot{A}$, which is updated from $\dot{A}$, the updated differential page, $\bar{A}_2$, is calculated by applying $A \odot \ddot{A}$, unless the update from $\bar{A}_1$ to $\bar{A}_2$ violates the unidirectional page programming constraints. Subsequent page updates up until $\bar{A}_4$ can be performed in an identical manner. During the updates, original pages can be also updated when the unidirectionality conflicts occur in the difference pages. However, the efficiency of the dual update scheme, the combination of the in-place update and difference update, directly depends on the number of '0' bits in the original page.

For example, to retrieve a page that has already been updated 4 times with the aforementioned mechanism, the most up-to-date version of the page, $\ddddot{B}$, can be obtained as follows: $\ddddot{B} = B \odot \bar{B}_4$, where B is the original page and $\bar{B}_4$ is the most recently updated difference page; the procedures also can be seen in figure 3. $\bar{B}_4$ contains the aggregated non-overlapping bit difference.

However, the page updates both in the data block and the difference block may cause bit screening violations due to the allowance of the random order page programming inside a block. Thus, the page updatability check should be examined before the actual update. Any '0' programmed bit (reset bit) in adjacent pages triggering a bit screening effect prohibits correct programming of the same bit of the page under programming. Consequently, all the programmed pages in a block whose page number is greater than the page under programming should be read and examined for possible bit screening. This scheme may cause significant overhead when a page to be programmed has many programmed pages that may introduce potential bit screening. Thus, limiting the maximum amount of neighbor page comparisons can effectively guarantee reasonable overhead of the updatability check. For this purpose, each page is

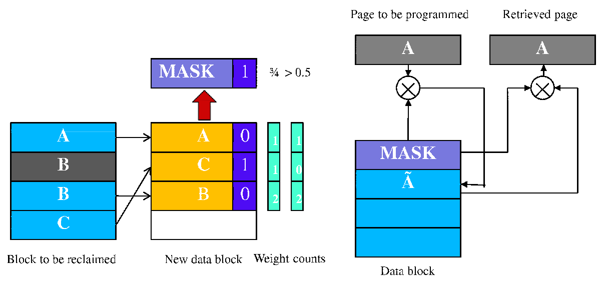

Figure 4: Block-Level Bit Masking and Reordering

augmented with additional *Updatability* information determined by the amount of updates that have already taken place in that page and by the distance between the page and the most distant programmed page.

4.2 Reduction of Bit Screening Through Block-Level Masking

As the pages in a data block are consumed by page writes and updates, the amount of available empty pages shrinks. When no empty page remains or no update is allowed for an upcoming write/update traffic, the block should be reclaimed for future use. The reclamation operation is comprised of two phases; (1) copying all the valid pages to a new block, and (2) erasing the block.

When valid pages are copied to a new block, full information on the pages such as bit contents and update counts can be obtained and utilized towards further optimizations. Consequently, we can perform extensive optimization in this phase to enhance page reuse by transforming the bit information inside a page and reordering the page locations. The page content transformation is beneficial when the transformed data has reduced the number of '0' bits, thus affording increased updatability. The page reordering process helps minimize the bit screening effect by incorporating page access patterns. In general, a well-tuned technique combining the two mechanisms is required to achieve maximum benefits.

The page content transformation can be easily performed by introducing a block-level bit mask in which the '0' bit in each column is counted and the associated mask bit is set when the number of '0' bits exceeds a given threshold. The overall number of '0' bits in a page can be reduced but page updatability may not be significantly improved due to possible bit screening by the preceding pages. Thus, the frequently updated pages are programmed last in sequential page program order and the weighted counts of '0' bits in each column are applied to decide the bit mask of the column as can be seen in figure 4.

A mask bit is set when $\sum_i Z_i / \sum_i W_i \cdot Z_i > \delta$, where W_i denotes the weight of the i^{th} page representing the amount of page updates before the reclamation, Z_i, the inverse of the bit on the i^{th} page, and δ, the threshold value to flip the bit. This scheme should be applied for all the 2K bits of the valid pages and the mask value can be determined and should be stored in the first page of the new block.

When a page is read, the read data also go through the masking process to revert the transformed data back to the original data. To simply achieve this transformation, the masking process applies a simple XOR operation both for reading and writing pages.

This scheme is only applicable to data blocks to enhance the in-place page update possibility. However, the difference block does not require the masking mechanism because it maintains already transformed data contents that are varying at run-time. To calculate a mask during a block reclamation, a NAND block-sized reclamation buffer and a physical NAND flash page per block should be allocated to store the block-level mask information. To further minimize the performance degradation of accessing masks, a small cache to hold the most recent block-level masks should be incorporated.

4.3 Effective Traffic Classification

As discussed in Section 3, the page access frequency can be highly skewed. Even in a single block, pages of highly varying frequency of access may cohabit, ranging from highly frequently accessed to quite infrequently accessed pages. While the mechanisms discussed in subsection 4.1 reduce the amount of page programming due to frequently accessed pages, the infrequently accessed pages in a block continuously produce unnecessary page writes when the block is reclaimed. Although these pages are not updated, they should be copied to the new block during reclamation because they still hold valid data. Consequently, the amount of infrequently accessed pages in a block directly affects block usage efficiency. The negative impact of this aspect is accentuated when the block also contains highly frequently accessed pages.

To minimize this impact, pages should be classified and spatially clustered based on their access frequency. In subsection 4.1, we proposed a two block scheme to maintain original data and the difference data. When a page update is required, the update should be preferably processed only in the difference block instead of utilizing both the data block and the difference block simultaneously. Consequently, data blocks should hold infrequently accessed pages and difference blocks should hold frequently accessed pages in the long run. When a difference block is reclaimed, the valid pages in the block should be moved back to the associated data block. Significant reduction in page programs during reclamations can be attained by utilizing this method of traffic classification.

4.4 Reduction of Cell Defect Probability

The durability of the NAND flash memory is directly affected by the amount of block erasures. Although a certain threshold of block erasures and page programs is guaranteed by manufacturers, the actual probability of the faulty cells due to these operations is predicated on the accumulated number of programmed '0' bits. The process of programming of a NAND flash cell causes physical damage to the thin oxide layer and the damage is accumulated over consecutive operations. As the voltage threshold window that separates the programmed state and the erased state nar-

Table 3: Spec2000 Benchmark Applications Summary

Application	Type	Code Size	Data Space
CRAFTY	Game	432KB	1150KB
GAP	Interpreter	912KB	1025KB
GCC	Compiler	1956KB	303KB
GZIP	Compressor	345KB	398KB
VPR	Circuit placement	79KB	1192KB

rows due to the accumulated damage, read sensing errors can be generated when the window excessively narrows, resulting in turn in drastic increases in run-time cell defect rate. For cell programming, the bits that are programmed to '0' experience high stress due to the applied high voltage. In addition to the reduction in the number of page programs through the advocated page reuse, the proposed mechanisms also reduce the total number of '0' bit programs to minimize the run-time probabilistic cell defects. Furthermore, bit disturbances caused by neighboring '0' programmed cells are also greatly relieved.

5. EXPERIMENTAL RESULTS

5.1 Experimental Framework

We have extensively utilized the SimpleScalar toolset to perform experiments under various conditions. The experimental architecture is equipped with two distinct 32KB direct mapped L1 I/D caches and a 128KB 4way set associative unified L2 cache with 2KB cache line size to efficiently interact with the NAND flash memory. To acquire the L1 miss penalty of the NAND based primary memory, we model the timing of the proposed architecture based on the operational timing of the SAMSUNG K9K8G08U0A NAND flash memory for the NAND access and the advanced SRAM for the interface with the processor. A setup comprised of a total 2GB of flash memory with two 8Gbit NAND flash memory is simulated. We perform simulation runs of 1 billion instructions per program. We simulate a set of representative programs from Spec2000, which approximate the behavior and complexity of real-life embedded applications. Table 3 outlines the benchmarks used for the experiments.

5.2 Expected Device Life-Cycle

The total number of replacement blocks and the consequent rate of block replacements directly impacts the life-cycle of the NAND flash memory device along with the distribution of the block replacements over the blocks. The proposed architecture significantly enhances page reuse frequency and associated block replacements. The experimental results observed in figure 5 demonstrate that the proposed architecture experiences on the average a 3.3 times reduction in block replacement compared to the baseline configuration, which already has achieved reductions in excess of 95% compared to the fixed block utilization. For *crafty*, *gap* and *gzip*, the proposed architecture shows more

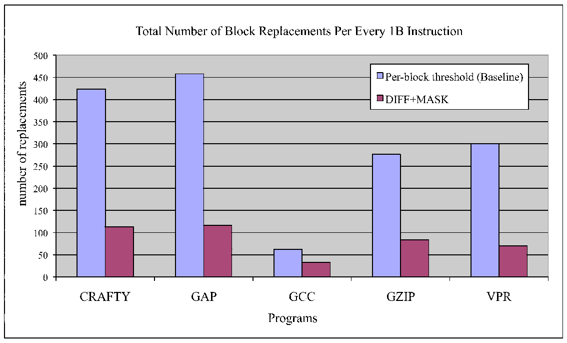

Figure 5: Number of Replacement Blocks per 1 Billion Instructions

than 3.5 times of page replacement reduction due to effective page reuse schemes. However, *gcc* shows a relatively low improvement. *Gcc* shows a significant amount of overlapping updates that cannot be handled by the proposed bit difference tracking scheme but traffic classification and block-level bit masking still seem to be beneficial. Furthermore, the proposed architecture minimizes the amount of '0' bits in programmed pages and the probability of bit defects caused by page programs can thus also be attenuated.

5.3 System Overhead

Compared to the baseline, the proposed architecture shows a slight performance degradation as can be seen in figure 6 even though the overall number of reclamations is significantly reduced. For a page read, it requires an additional XOR operation. If a page is updated, additional time to read the difference page with additional XNOR operations is required. However, both page reads from the data block and the difference block can be accessed simultaneously if these blocks are located in physically distinct NAND flash chips. For a page write, additional updatability checks reading multiple pages depending on the updatability of the preceding pages are required. However, the impact of the system level performance degradation is very limited and further alleviated when parallel NAND flash memory accesses are employed.

The proposed architecture highly relies on the optimal

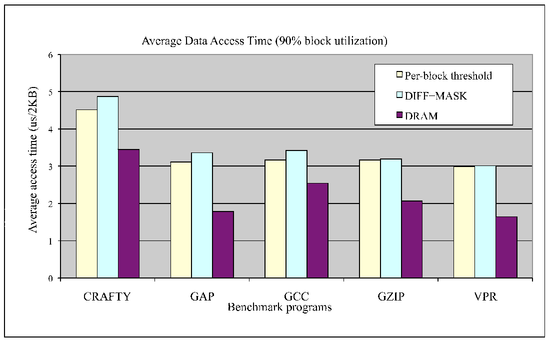

Figure 6: Average Access Time Comparisons

page reuse and associated reduction of block replacements. The block utilization ratio is determined based on the application behavior as in the previous work. Additionally, each block should consume one page to store block-level mask information. Blocks that are updated also require additional difference blocks but the total amount of difference blocks can be controlled. Along with this space overhead, the proposed architecture exhibits some degree of overhead in its logic and internal memory. Under the current experimental configuration, it consumes 256KB+64KB of SRAM for the read cache/write buffer and reclamation buffer. It further requires a small SRAM space for maintaining information in the block-level mask cache and additional housekeeping information. The logic overhead is mainly dedicated to the cache/memory steering logic and binary tree based address controlling logic. The overhead of additional XOR/XNOR logic is insignificant and the degree of the memory and logic overhead is already quite small, with its importance slated to diminish further in the near future as semiconductor technology continues to evolve.

6. CONCLUSION

In this paper, we have proposed a highly effective NAND flash memory page reuse mechanism to achieve high durability by aggressively applying a partial programming capability. The proposed architecture provides unique mechanisms to alleviate the page programming and associated block reclamation overhead through the reduction of block reclamations. Furthermore, the overall stress on each cell during page programming is significantly reduced by transforming the bit representations of the valid data into new bit streams. The benefit of the cell stress reduction may not be observed directly but the impact on the endurance of the NAND flash based memory could be significant. We believe that the proposed mechanisms can be applied to all NAND flash based memory systems and provide significant enhancement on the durability of the memory directly and indirectly. The experimental results show that the proposed architecture significantly prolongs the device life-cycle by minimizing the *program/erase* operations with slight degradation on average memory access time.

7. REFERENCES

[1] OneNAND^{TM}. http://www.samsungsemi.com.

[2] International Technology Roadmap for Semiconductors, 2009. http://www.itrs.net.

[3] Ahn, S., Song, Y., Jeong, C., Shin, J., Fai, Y., Hwang, Y., Lee, S., Ryoo, K., Lee, S., and Park, J. Highly manufacturable high density phase change memory of 64Mb and beyond. In *IEDM Technical Digest '04: Proceedings of the IEEE International Electron Devices Meeting* (2004), pp. 907–910.

[4] Grupp, L. M., Caulfield, A. M., Coburn, J., Swanson, S., Yaakobi, E., Siegel, P. H., and Wolf, J. K. Characterizing flash memory: anomalies, observations, and applications. In *MICRO 42: Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture* (2009), pp. 24–33.

[5] In, J., Shin, I., and Kim, H. SWL: a search-while-load demand paging scheme with NAND flash memory. In *LCTES '07: Proceedings of the 2007 ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools* (2007), pp. 217–226.

[6] Joo, Y., Choi, Y., Park, C., Chung, S. W., Chung, E., and Chang, N. Demand paging for OneNAND^{TM}Flash eXecute-in-place. In *CODES+ISSS '06: Proceedings of the 4th International Conference on Hardware/Software Codesign and System Synthesis* (2006), pp. 229–234.

[7] Kaufman, A. B. An expandable ferroelectric random access memory. *IEEE Transactions on Computers 22*, 2 (1973), 154–158.

[8] Lee, J., Lee, C., Lee, M., Kim, H., Park, K., and Lee, W. New programming disturbance phenomenon in NAND flash memory by source/drain hot-electrons generated by GIDL current. In *NVSMW '06: Proceedings of the 21st Non-Volatile Semiconductor Memory Workshop* (2006), pp. 31–33.

[9] Lee, K., and Orailoglu, A. Application specific low latency instruction cache for NAND flash memory based embedded systems. In *SASP '08: Proceedings of the 2008 Symposium on Application Specific Processors* (2008), pp. 69–74.

[10] Lee, K., and Orailoglu, A. Application specific non-volatile primary memory for embedded systems. In *CODES+ISSS '08: Proceedings of the 6th IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis* (2008), pp. 31–36.

[11] Masuoka, F. New ultra high density EPROM and flash with NAND structure cell. In *EDM Technical Digest '87: Proceedings of the IEEE International Electron Devices Meeting* (1987), pp. 552–555.

[12] Park, C., Lim, J., Kwon, K., Lee, J., and Min, S. L. Compiler-assisted demand paging for embedded systems with flash memory. In *EMSOFT '04: Proceedings of the 4th ACM International Conference on Embedded Software* (2004), pp. 114–124.

[13] Park, C., Seo, J., Bae, S., Kim, H., Kim, S., and Kim, B. A low-cost memory architecture with NAND XIP for mobile embedded systems. In *CODES+ISSS '03: Proceedings of the 1st IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis* (2003), pp. 138–143.

[14] Tehrani, S., Slaughter, J. M., Deherrera, M., Engel, B. N., Rizzo, N. D., Salter, J., Durlam, M., Dave, R. W., Janesky, J., Butcher, B., Smith, K., and Grynkewich, G. Magnetoresistive random access memory using magnetic tunnel junctions. In *Proceedings of the IEEE* (2003), pp. 703–714.