Application Specific Non-Volatile Primary Memory for Embedded Systems

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

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

ABSTRACT

Memory subsystems have been considered as one of the most critical components in embedded systems and furthermore, displaying increasing complexity as application requirements diversify. Modern embedded systems are generally equipped with multiple heterogeneous memory devices to satisfy diverse requirements and constraints. NAND flash memory has been widely adopted for data storage because of its outstanding benefits on cost, power, capacity and non-volatility. However, in NAND flash memory, the intrinsic costs for the read and write accesses are highly disproportionate in performance and access granularity. The consequent data management complexity and performance deterioration have precluded the adoption of NAND flash memory. In this paper, we introduce a highly effective non-volatile primary memory architecture which incorporates application specific information to develop a NAND flash based primary memory. The proposed architecture provides a unified non-volatile primary memory solution which relieves design complications caused by the growing complexity in memory subsystems. Our architecture aggressively minimizes the overhead and redundancy of the NAND based systems by exploiting efficient address space management and dynamic data migration based on accurate application behavioral analysis. We also propose a highly parallelized memory architecture through an active and dynamic data redistribution over the multiple flash memories based on run-time workload analysis. The experimental results show that our proposed architecture significantly enhances average memory access cycle time which is comparable to the standard DRAM access cycle time and also considerably prolongs the device life-cycle by autonomous wear-leveling and minimizing the *program/erase* operations.

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

General Terms: Performance, Design, Experimentation

1. INTRODUCTION

A disproportionate hardware cost in most embedded systems, especially in consumer electronic systems, is incurred by the memory subsystem as a variety of different memory components are concurrently adopted. The memory requirements of embedded systems are far more diverse and constrained than those of generic computer systems. DRAM has long been adopted as a single and universal standard memory device in generic computer systems. In contrast, there exists no single or universal memory device for embedded systems. Instead, highly customized memory devices inhabit the space of current offerings. Embedded systems are defined by their own individual requirements and constraints and furthermore, these characteristics tend to grow more complicated and diversified in future generations. The memory subsystem plays a critical role in embedded systems under the aforementioned technological trend and more emphasis should be placed on this in the near future.

Unification of memory devices in embedded systems is beneficial in many ways. It generally guarantees low cost, low power consumption and design ease for the systems. Furthermore, attainment of the highest benefits necessitates that the memory possess the capability of direct interfacing with the processor and that it fulfill specific non-volatility requirements. Most embedded system designers anticipate such non-volatility requirements as they can drastically ease their ultimate memory management burden when power issues are incurred.

NAND flash memory has long been considered and adopted in practice as the most preferable data storage device since it is suitable for embedded systems which generally require special characteristics such as low cost, low power, high capacity, non-volatility, and high reliability. Therefore, NAND flash memory is highly appealing as a secondary or tertiary memory due to several beneficial characteristics such as large granularity of access and high sustained performance. However, the random-bit inaccessibility, the large and diverse granularity of accesses, and the long and imbalanced access latency make it difficult for it to be adopted as a primary memory. A significant amount of research has been undertaken to overcome these limitations in utilizing NAND flash memory as a unified primary memory and many researchers still envision NAND flash memory as the most plausible candidate for this purpose.

In this paper, we introduce a novel unified non-volatile primary architecture based on NAND flash memory. Embedded systems are generally composed of a fixed set of functionalities and exhibit a great degree of application specific behaviors [7, 8]. This fact encourages many researchers to exploit application information in a variety of ways. There have been two most popular approaches in employing application specific information. One consists of transforming and relocating instructions/data to exhibit enhanced runtime behavior. This information is gathered off-line from the given set of applications with the help of compilers or linkers. The other is extracting application specific information from a given set of applications and tuning the hardware based on the obtained information. We mainly utilize the latter approach in this work as it offers increased accuracy and efficiency at run-time. We introduce a novel primary memory architecture that demonstrates non-volatility, highly optimized and balanced average access time both in read and write operations, and great enhancement in performance by exploiting parallelism in multiple devices. Extensive simulation using the SimpleScalar simulator with Spec2000 benchmark suites illustrates the significant advantages of our architecture over the generic DRAM or other baseline NAND-based architectures.

The rest of the paper is organized as follows. Section 2 briefly presents NAND flash memory based memory systems. Section 3 delivers background and the basic technical motivation. Section 4 introduces our non-volatile primary memory architecture and the techniques of dynamic data/address management, aggressive multi-level data migration and multi-device optimization. Section 5 presents experimental results based on simulation and provides a comparison with the baseline memory architecture. Section 6 provides an overall summary of the proposed architecture.

2. RELATED WORKS

Due to the increasing requirements and complexity in embedded systems, memory subsystems have undergone diversification and increased complexity. A significant amount of research has been performed in the field to unify memories or reduce the design complexity of the memory systems by employing non-volatile memory, especially NAND flash memory, with aggressive exploitation of application specific information.

Over a decade, a considerable amount of research on the NAND flash based memory systems has been introduced. Park et al. in [6] 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. To minimize access latency for time critical applications, they prioritize applications based on their criticality. However, this system requires a relatively large cache and it suffers from overall performance degradation. In [3], OneNAND$^{\mathrm{TM}}$, a hybrid NAND flash memory, has been introduced as a high-performance non-volatile memory for embedded systems. In [5, 2, 1], the authors develop NAND flash memory based code storage systems by incorporating the VM (Virtual Memory) subsystem of the processor with no need for hardware modifications. However, these approaches are limited to certain applications and require considerable amount of VM page buffers.

3. BACKGROUND AND MOTIVATION

Memory is one of the most critical components in embedded systems but the performance discrepancy between the processor and memory widens as technology evolves. Technological advancements in memory mostly emphasize capacity enlargement complemented by steady but nonetheless meager performance enhancements. Most systems including embedded systems try to relieve the performance problem by introducing an efficient hierarchical memory architecture from cache memory down to secondary/tertiary memory. However, in embedded systems, this standard memory hierarchy complicates the system design from the vantage point of cost, size and power due to the introduction of multiple heterogeneous memory components. To unify memory components in embedded systems, a superior primary memory device whose characteristics satisfy every design requirement should be present.

From the perspective of performance, access to memory is one of the most critical and time consuming processes in the execution of code in the processor. While SRAM/DRAM demonstrates highly superior access behavior both in the *preload* time and the *transfer* time, NAND flash memory demonstrates a significantly long access time in terms of both read and write for the predefined access unit of a page as can be seen in Table 1. Yet a sharp contrast can be drawn between these two operations in the NAND flash memory, as reading a page is performed by sensing the voltage levels of the given array, thus being relatively fast, while in contrast, writing a page is non-atomic and composed of multiple low-level operations, thus consuming a huge amount of time. Based on the NAND flash memory architecture, each cell in a page can change its bit status by charging or discharging their floating gate; the charging and discharging operations cannot be performed though concurrently, which induces unidirectionality in bit transitions. In NAND flash memory, there exist two distinct operations, *erase and program*, based on the direction of bit transitions. The logical write operations can be expressed by combining both erase and program operations and furthermore those operations are extremely time-consuming. The asymmetric nature of the NAND flash memory operations raises a lot of problems and discourages the adoption of the NAND flash memory. In addition, the discrepancy between read and write operations in access time increases the design complexity of the NAND flash memory based systems.

From an architectural point of view, most embedded processors incorporate a small size of instruction and data cache. In general, the hit rate of these caches is substantially high and the average cache miss time for the standard primary memory, represented as the product of cache miss rate and penalty, is maintained sufficiently low compared to the cache hit time. However, for the long latency memory device like the NAND flash memory, a relatively low cache hit rate also results in unacceptably high average memory access times due to the extremely high penalty on cache misses. Furthermore, the performance asymmetry in NAND flash memory produces more critical issues. Consequently, a fair amount of research to lessen or hide the latency of the NAND flash memory based embedded systems has been initiated. Along with the performance issues, the NAND flash memory is unable to update its page in place due to the intrinsic "unidirectionality" in changing bits. Updating a page will generally produce obsolete pages and the consequent introduction of redundancy negatively impacts the performance of the NAND flash memory. Memory is traditionally conceived as a passive component in computer systems. Most

Table 1: Timing Characteristics of Memory Devices

Device	SRAM	DRAM	NAND FLASH	OneNAND^{TM}
Access Time (ns)	5	50	30000	25000
Read Bandwidth (MB/s)	2500	1000	17	108
Write Bandwidth (MB/s)	2500	1000	6.8	9.3

research in memory is conducted to relieve the overhead to the memory by manipulating memory accesses in the upper layer [4]. However, these techniques do not fit well to the NAND flash memory whose underlying characteristics diverge greatly from standard memory. Consequently, an active memory can efficiently utilize their storage space by dynamically processing the full range of information they have acquired at run-time. Whenever data is transferred to the memory, it provides a lot of meaningful information: access behavior and frequency, and data types. The access behavior and frequency can be fully utilized to reorganize the physical data layout in the memory to reduce conflicts. Data types may help in identifying access conflicts which in general cause cache misses and result in long latency memory accesses.

Data accesses to memory chunks (pages) generally exhibit diverse access frequency and patterns as can be seen in Fig 1. This may cause a significant degradation in the performance of the NAND flash memory based systems. If rarely accessed pages are defined as cold and frequently accessed pages as hot in their access temperature, most applications demonstrate a highly skewed statistical distribution on the access temperature. Access patterns exhibit variability over time enabling such change information to be dynamically captured to optimize memory behavior by adapting memory organization according to their run-time information.

The highly skewed access behavior and high run-time variability in access patterns have long been an issue in the research area of NAND flash memory. While most previous research on NAND flash memory has focused on controlling local or global redundancy without knowledge of the run-time behavior of the application, an active memory can effectively balance the access skew and dynamically adapt itself to the run-time variability by introducing the internal hybridization of memory, which in turn enables memory to systematically synthesize individually preprocessed information from internal components of the memory and more accurately control the fine-grained redundancy.

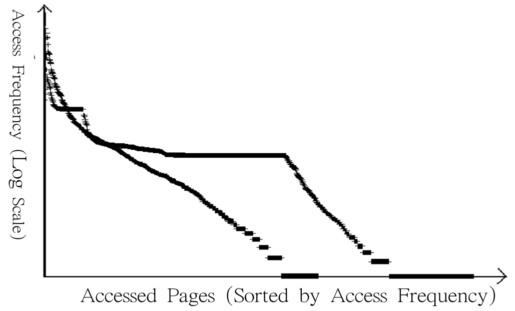

Figure 1: Example of Statistical Access Distribution of Different Applications

4. PROPOSED ARCHITECTURE

Primary memory basically constitutes the first level of off-chip memory hierarchy and its performance and characteristics greatly impact the performance of the complete system. Traditionally, primary memory exhibits reasonable access latency both in read and write operations without any performance differentiation. To achieve the equivalent performance in the NAND flash memory based primary memory, the characteristics and behaviors of the system especially with regard to the memory subsystem should be accurately identified, thus, enabling the aggressive utilization of the derived information so as to optimize the subsequent behavior of the memory.

Most standard memory systems find a way to relieve performance degradation by introducing multiple levels of memory hierarchy. This will help if a higher degree of locality can be captured in the upper memory layers. As we observed in Section 3, a frequently accessed group of pages can be absorbed by introducing a level of internal cache/buffer inside memory. Along with the fast internal cache, the hardware interface to the standard memory of the proposed primary memory should be equivalently fast. Furthermore, the aforementioned employment of the application specific information can be processed in an *application-specific control logic.* The novel ideas of the proposed memory architecture lie in the fair and optimal resource allocation and usage within a dynamically variable execution environment in conjunction with an accurate understanding of the behavior of accesses to memory through the internal hybridization of memory. The detailed architectural ideas are introduced in the subsequent subsections; the overall hardware architecture can be seen in Fig 2.

4.1 Application Specific Dynamic Memory Management

Physically, a number of pages constitute a block that is the basic unit for erasure and fundamental unit for data management in NAND flash memory. In most NAND flash memory based systems, redundancy is unavoidable and incorporated

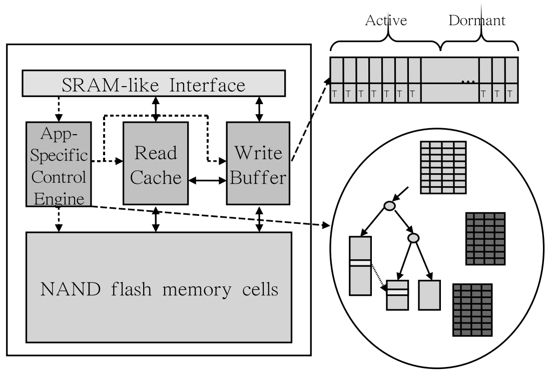

Figure 2: The Proposed Memory Architecture

either locally or globally to prevent excessive data overflow resulting in block replacements. A simple fixed local threshold for each block can be proposed to relieve the data overflow. However, it causes highly biased accesses that are extremely undesirable in the flash memory. Global overflow buffers can be alternatively employed but they would result in excessive data migration and inefficiency in the level of redundancy.

We propose a dynamic threshold on a per-block basis, *the valid page threshold*, which controls the block replacement cycles through incorporation of application specific access frequency. Initially, the access temperature, degree of access frequency, and average access frequency can be acquired through the profiling of the given application. Then, based on this information, a proper valid page threshold for each block can be computed according to the following three constraints, *(1)* $0 \leq Th_{B_i} \leq Th_{init} \leq \alpha$ *(2)* $\sum_i Th_{B_i} = \beta$ *and* $i \leq \gamma$ *(3) minimize* $\sum_i (F_i - (F_{avg} \times i))$, where α represents the total number of pages in a block in the NAND flash memory, β the number of pages consumed by the application and γ the total number of blocks in the flash memory. Page clustering generally reduces the number of block replacements by properly allocating overflow buffers according to their average temperature and also facilitates the wear-leveling of the blocks by evenly controlling the rate of block replacements for each block. Furthermore, the temperature information is fed back to the write buffer to help the page replacement scheme. As can be seen in Fig 2, the write buffer is composed of two regions: active and dormant based on their access temperature. Only pages in the dormant region are chosen as a replacement page, which provides a more accurate replacement policy in the buffer and consequently guarantees fewer buffer misses.

Furthermore, the average time to fill the overflow buffer is already known and can be utilized to hide the times for the block replacement. We assume that the rate of accesses to the block i is constant and defined as t_i. Then the total amount of time spent to fill the overflow buffer can be expressed as $t_i^B F = \sum_{i=Th_i}^{\gamma} t_i$. If $t_i^B F$ is comparatively long, the block replacement operation can be performed while filling the overflow buffers. Thus, another threshold, *the reclaim initiation threshold*, is introduced to initiate in a timely manner block replacement to hide the overhead of the redundant operations. A value for this threshold can be obtained by applying the following formulae; *(1)* $Th_{B_i} \leq Th_{B_i}^{REC} \leq \gamma$ *(2)* $t_i^B F \equiv \delta$, where δ denotes the time to program a page in the NAND flash memory and is provided as a constant value in the NAND flash memory specification. The timing diagram for the proposed algorithm can be seen in Fig 3. Properly applied, most of the time spent on block replacement can be effectively overlapped with the time it takes to fill the overflow buffer.

4.2 Efficient Address Generation and Data Migration

Based on the thresholds that are described in the previous subsection 4.1, the physical memory space in the NAND flash memory is divided into non-uniformly sized contiguous memory chunks delimited by the threshold of each block. However, the temperatures of blocks may change over time and the patterns of temperature changes of blocks are not unique. These changes can be captured only at run-time and dynamic threshold adjustment techniques need to be

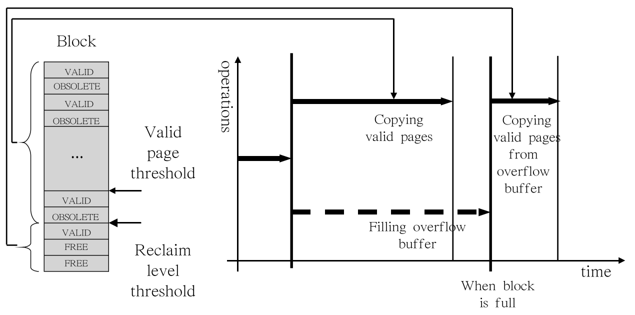

Figure 3: Timing Diagram for Block Replacement Based on Thresholds

employed. The dynamic threshold adjustment introduces two additional challenges. Firstly, the address space may no longer be contiguous as pages in the middle of a block may be migrated. Secondly, it complicates the address generation mechanism. To begin with, we categorize the temperature changes and resulting address changes as two distinct types. The type 1 block, whose temperature changes from cold to hot, contains fewer overflow buffers than it requires. Instead of expanding the overflow buffer, we migrate away pages based on the following criteria. Firstly, the coldest page should be chosen to minimize the negative impact of the temperature rise at the target block. Secondly, higher priority is given to the pages at the edge to facilitate address generation by keeping address contiguity. A type 2 block, whose temperature has undergone changes from hot to cold, exhibits more space for the valid page to accommodate and mark that block as "borrowable". By applying local data migration based on information gathered, the levels of overflow buffers of blocks are adjusted dynamically and provide more effective block replacements. However, from the perspective of address generation, it raises two subsequent problems. Firstly, the logical address space needs to be mapped to non-uniformly sized physical addresses. Secondly, random data migration between non-uniformly sized blocks needs to be allowed. In representing a non-uniformly sized physical address space, a B-tree can be a good candidate because it exhibits easy address space management, fast address generation time, and dynamic address space adjustment. However, random data migration of arbitrary leaf nodes in the traditional B-tree can readily deteriorate the aforementioned intrinsic characteristics of the B-tree. Consequently, we propose an extended B-tree that permits local data migration in which source and destination blocks share a common ancestor within a pre-defined depth. A hashing based extra addressing scheme hybridized with the original B-tree is introduced to achieve a fast and efficient data migration. Further optimization can be effectively accomplished by simplifying migration links by removing cascaded links and circular links as in Fig 4. The proposed hybrid scheme provides predictable and flexible address generation while bounding both the hashing hardware cost and indirect traversal length.

In principle, the basic data migration is performed by utilizing local block behaviors. Further global data migration can accelerate the efficient utilization of the limited page/block resources with more accurate global temperature redistribution. This process can be performed in background by continually searching and migrating proper blocks to fit the following criteria. A *mergeable block* set is composed of

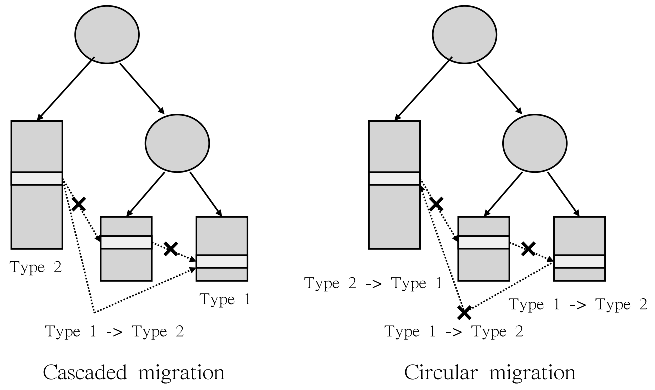

Figure 4: Techniques for Simplifying Migration Links

two or more adjacent blocks that experience steady temperature changes from hot to cold, while the total number of valid pages of all the candidate blocks cannot exceed the number of pages in a block. On the other hand, the *separable block* is defined as a block that experiences steady temperature change from cold to hot, and no local data migration. As can be seen in Fig 5, we can retrieve over-allocated pages and redistribute those free pages/blocks to highly active blocks by providing an efficient coarse grained global data migration.

4.3 Exploiting Parallelism in Array of Memories

As technology evolves, the cost of NAND flash memory continues to diminish but the performance of NAND flash memory remains near constant. Thus, there has been considerable amount of research on high performance massive storage device design incorporating an array of NAND flash memory to overcome this problem. By introducing and exploiting parallel access in multiple NAND flash memories, we logically can attain performance enhancement proportional to the total number of flash memories accessed in parallel. However, in reality, without proper arbitration of the data traffic to the devices, the full degree of parallelism cannot be exploited. To avoid constant conflicts in accessing the same physical device, we propose data reorganization and redistribution over multiple devices based on their temperatures.

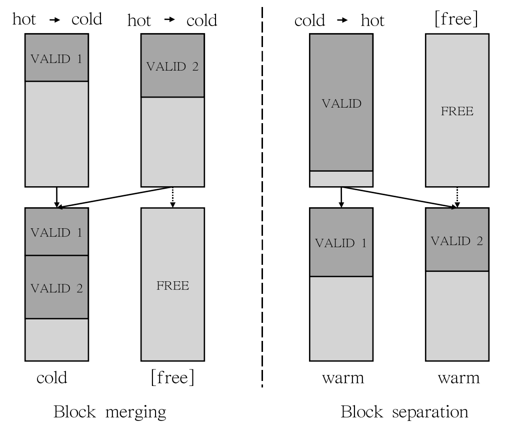

Figure 5: Techniques for Redistributing Pages by Merging and Separation

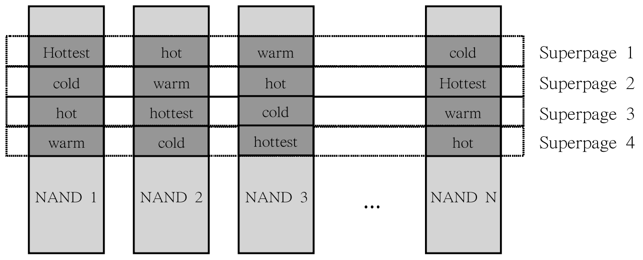

Figure 6: Data Layout to Minimize Access Conflicts by Redistributing Pages

A superpage and data layout to minimize single device access conflicts by redistributing pages are depicted in Fig 6. The basic ideas lie in the fact that the average temperature in different physical devices should be balanced and furthermore, the distance between the hottest pages in a block be widened to prevent single device access conflicts with the help of maximally parallel write-backs from the write buffer.

5. EXPERIMENTAL RESULTS

5.1 Experimental Framework

We extensively used the SimpleScalar toolset to perform our experiments under various conditions. Our experimental architecture is equipped with two distinct 32KB direct mapped L1 I/D caches with 2KB cache line size to efficiently interact with the NAND flash memory. The size of the read cache and write buffer of the proposed architecture is 64KB and both have 2KB cache lines. A 200MHz core with a consequent 5ns clock cycle is assumed. We also assume 18 cycles for an L1 cache miss on the DRAM. 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. We performed simulation runs of 100 million instructions per program with the initial thresholds computed after running each program. We simulate a set of representative programs from Spec2000, which approximate the behavior and complexity of real-life embedded applications. Table 2 outlines the benchmarks used for the experiments.

5.2 Average Memory Access Time for Data

As mentioned in Section 3, the disproportionate access times in read and write operations in NAND flash memory result in serious performance degradations. Our proposed memory architecture exhibits great performance enhancement over generic flash memory solutions utilizing fixed and global threshold algorithms by minimizing the write-back operations and furthermore, efficiently hiding the time overhead of block replacements. The average data access time

Table 2: 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

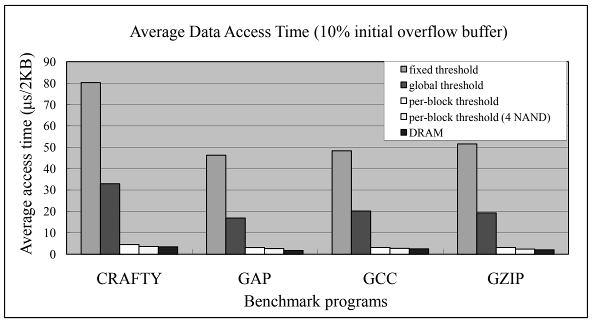

Figure 7: Average Access Time Comparisons

for a 4-NAND based memory architecture is as low as the access time of DRAM as can be observed in Fig 7. All the benchmark programs show similar patterns of results. The average data access time with the proposed architecture is less than 1/15 of the passive NAND flash memory solution, and slightly longer, generally up to 16%, than a DRAM based system with the exception of the GAP program that shows higher data access time due to a higher rate of write-back operations as conflicts in the data buffer grow.

5.3 Expected Device Life-Cycle

The content validity of each NAND flash memory block is only guaranteed up to 100,000 program/erase cycles. Thus, the total number of replacement blocks and subsequent rate of block replacement directly impacts the life-cycle of the NAND flash memory device along with the distribution of the block replacements over the blocks. The experimental results observed in Fig 8 demonstrate that the proposed architecture experiences less than 95% and 80% reduction in block replacement compared to fixed threshold and global threshold approaches respectively. Furthermore, the proposed architecture has autonomously distributed the block replacement evenly due to the dynamic adjustment of the size of overflow buffers. The overall expected device cycles can be considerably prolonged accordingly.

5.4 System Overhead

The proposed architecture highly relies on the optimal re-distribution of redundancy in the memory. As can be seen in subsection 5.2, the system-wide redundancy should be determined based on the requirements of the system. 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 128KB of SRAM for the read cache and write buffer. It further requires SRAM space for maintaining information in the extended B-tree. The logic overhead is mainly dedicated to cache/memory steering logic and B-tree based address controlling logic. However, the degree of the memory and logic overhead is already reasonably 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 non-volatile primary memory architecture incorporating NAND flash memory to provide a high-performance and low-cost memory solution. The proposed architecture provides a unified non-volatile primary memory solution which relieves the

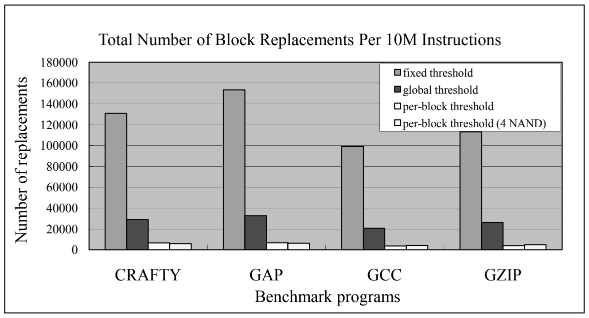

Figure 8: Number of Replacement Blocks per 10 Million Instructions

growing design complexity in memory subsystems. We have designed a hybrid memory architecture by intelligently incorporating application specific information to provide optimal physical data layout on the NAND flash memory. We further provide an efficient local and global dynamic data migration and flexible address generation mechanism to effectively transform the data layout based on the dynamic access behavior changes. All the redundant operations including block replacement and free block preparation are performed so as to minimize their negative impact on the performance. The experimental results show that our proposed architecture significantly enhances average memory access time and also extensively prolongs the device life-cycle by minimizing the *program/erase* operations and by autonomous wear-leveling.

7. REFERENCES

[1] J. In, I. Shin, and H. Kim. 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*, pages 217–226, 2007.

[2] Y. Joo, Y. Choi, C. Park, S. W. Chung, E. Chung, and N. Chang. 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*, pages 229–234, 2006.

[3] B. Kim, S. Cho, and Y. Choi. OneNAND^{TM}: A high performance and low power memory solution for code and data storage. In *Proceedings of 20th Non-Volatile Semiconductor Workshop*, 2004.

[4] P. R. Panda, F. Catthoor, N. D. Dutt, K. Danckaert, E. Brockmeyer, C. Kulkarni, A. Vandercappelle, and P. G. Kjeldsberg. Data and memory optimization techniques for embedded systems. *ACM Transactions on Design Automation of Electronic Systems*, 6(2):149–206, 2001.

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

[6] C. Park, J. Seo, S. Bae, H. Kim, S. Kim, and B. Kim. 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*, pages 138–143, 2003.

[7] P. Petrov and A. Orailoglu. Energy frugal tags in reprogrammable I-caches for application-specific embedded processors. In *CODES '02: Proceedings of the Tenth International Symposium on Hardware/Software Codesign*, pages 181–186, 2002.

[8] P. Petrov and A. Orailoglu. A reprogrammable customization framework for efficient branch resolution in embedded processors. *ACM Transactions on Embedded Computing Systems*, 4(2):452–468, 2005.