Exploiting Narrow-Width Values for Improving Non-Volatile Cache Lifetime

Guangshan Duan and Shuai Wang
State Key Laboratory of Novel Software Technology
Department of Computer Science and Technology
Nanjing University, Nanjing, Jiang Su, China
Email: guangshan_duan@smail.nju.edu.cn, swang@nju.edu.cn

***Abstract*—Due to the high cell density, low leakage power consumption, and less vulnerability to soft errors, non-volatile memory technologies are among the most promising alternatives for replacing the traditional DRAM and SRAM technologies used in implementing main memory and caches in the modern microprocessor. However, one of the difficulties is the limited write endurance of most non-volatile memory technologies. In this paper, we propose to exploit the narrow-width values to improve the lifetime of non-volatile last level caches. Leading zeros masking scheme is first proposed to reduce the write stress to the upper half of the narrow-width data. To balance the write variations between the upper half and the lower half of the narrow-width data, two swap schemes, the swap on write (SW) and swap on replacement (SRepl), are proposed. To further reduce the write stress to non-volatile caches, we adopt two optimization schemes, the multiple dirty bit (MDB) and read before write (RBW), to improve their lifetime. Our experimental results show that by combining all our proposed schemes, the lifetime of non-volatile caches can be improved by 245% on average.**

I. Introduction

Although DRAM and SRAM are widely used for building blocks in main memory and on-chip caches in the modern microprocessors, recent research has focused on finding the new technologies to replace the DRAM and SRAM designs due to their increasing problems in cell density, leakage power, and reliability against soft errors. The emerging non-volatile memory technologies, such as phase change memory (PCM), spin-torque transfer RAM (STTRAM), and resistive RAM (ReRAM), are among the most promising technologies for future memory and cache designs.

Besides their no-volatility, the non-volatile memory technologies can avoid the refreshing of the memory cell in the DRAM design, and also reduce the leakage power. They can also provide higher cell density compared to the traditional DRAM and SRAM technologies. For the reliability, the non-volatile memory is more robust against particle-induced soft errors. However, compared to traditional DRAM and SRAM, the main disadvantages for the non-volatile memory are higher write latency, higher write energy, and less write endurance. Especially if we want to adopt the non-volatile memory technology at the on-chip cache level, the write endurance will become worse compared to the lower level memory hierarchy, such as main memory. The high frequency of write operations on caches will reduce the lifetime of the non-volatile caches dramatically. Although recent work has proposed some wear leveling schemes to increase the lifetime of the non-volatile main memory [1][2][3], we cannot directly apply their schemes to the on-chip caches due to the different operational behaviors between main memory and on-chip caches. Therefore, the techniques targeting at improving the lifetime of the non-volatile on-chip caches are needed.

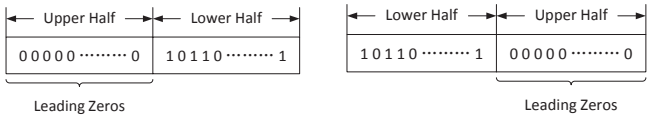
In this work, we propose the microarchitectural level solution to improve the lifetime of the non-volatile last level cache (LLC) by exploiting the narrow-width values. The presence of narrow-width values in last level cache is first studied and a leading zeros masking (LZM) scheme is proposed. To mitigate the write variation on our LZM design, two swap schemes, the swap on write (SW) and swap on replacement (SRepl), are proposed to balance the write operations to the upper and lower halves in our narrow-width data. To further improve the lifetime of our non-volatile on-chip cache, we propose to adopt two optimization schemes, the multiple dirty bit (MDB) and read before write (RBW), to reduce the unnecessary writes to the cache. Our experimental results indicate that our LZM with swap scheme can improve the lifetime by 33.1%, and by further applying the MDB and RBW schemes, the lifetime improvement rate of our non-volatile LLC will increase to 245% on average.

The rest of the paper is organized as follows. In the next section, we discuss related work for non-volatile memory and cache designs. In Section III, we provide details on our proposed cache lifetime improvement schemes by exploiting the narrow-width values. We present our experimental setup and results in Section IV. Section V draws the conclusion and discusses the future work.

II. Related Work

To improve the lifetime of the non-volatile memories, there are mainly three types of solutions: a) Write Reduction [1][4][5], which targets at reducing the amount of writes or bit flips in the memory cells. b) Write Variation Balancing (Wear Leveling) [2][6][7], which tries to even out the unbalanced write frequencies to all memory lines or cachelines. c) Error Correction [8][9][10], which uses redundancy or error correction coding schemes to extend the lifetime of the memory by fault tolerance. However, most of the previous work was focused on improving the lifetime of non-volatile main memory, not for on-chip caches. [6] extended the wear leveling schemes from the main memory to the on-chip caches and [7] focused on mitigating the cacheline-level write variations. Nevertheless, there is no scheme targeting at balancing the

978-3-9815370-2-4/DATE14/©2014 EDAA



(a). Narrow Width (b). After Swap

Fig. 1. Narrow-width values (leading zeros) in the cache.

write variation within the cacheline. Our proposed design focuses on word-level write reduction and variation balancing within the cacheline by exploiting the narrow-width values, and can be combined with other cacheline-level designs and error correction coding schemes to further improve the lifetime of the non-volatile on-chip caches.

III. Narrow Width Value for Lifetime Improvement

A. Narrow Width Value

The narrow-width values in high-performance microprocessors have been well studied and exploited for performance, power, and reliability optimizations in register files and caches [11][12][13]. In general, values with leading '0's are referred to as narrow-width values. Therefore, we can only store portion of the data without the leading '0's and restore it by the zero-extension logics when reading it out. Figure 1 (a) shows an example of the narrow width value we defined in this work. First, we divide the data into two halves, the lower half and the upper half. If the upper half of the data are all zeros, we define this type of data as narrow-width data. Therefore, when we write this type of data into the cacheline, the write operation of the upper half bits can be avoided, and thus the write stress to the cacheline can be reduced. We call this scheme, the leading zeros masking (LZM) scheme, which has been used in energy reduction in traditional cache design [14]. Note that the data unit we mentioned here can be a byte, 16 bits, 32 bits, or 64 bits in the cacheline. If we choose the byte-level study for the narrow-width data, the percentage of the narrow-width data in the cache may be high. However, the area overhead will be also high due to the extra bit N (1/8 = 12.5%) needed for indicating whether the data is narrow-width or not. Therefore, we choose the 64-bit as the data unit in our following study for its low area cost.

B. Swap for Write Variation Balancing

The leading zeros masking scheme can reduce the number of writes to the upper half of the narrow-width data in the cacheline and thus improve its lifetime. However, the number of writes to the lower half of the narrow-width data will remain the same. Therefore, the lifetime of the non-volatile cells in the lower half will not be improved. In order to further improve the lifetime of the lower half, we propose to swap the data between the lower half and the upper half periodically based on certain strategies. Figure 1 (b) shows the narrow-width data after the swap. If proper time intervals for the swap can be chosen, the write stress to both halves will be reduced and the write variations between two halves will be well balanced. However, if a very frequent interval is chosen, the overhead for the swap will be high. If we choose a very large swap interval, the write variations between two halves will not be well balanced. Therefore, we propose two strategies to do the swap: the swap on write (SW) and the swap on replacement (SRepl).

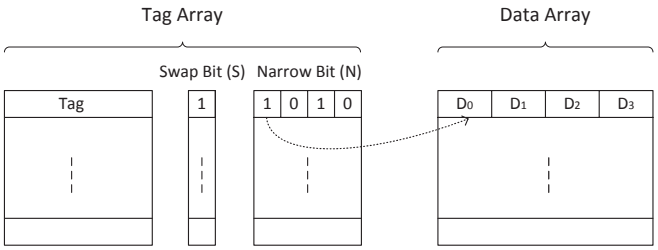


Fig. 2. Block diagram of the proposed non-volatile cache design.

1) Swap on Write: For this strategy, we do our swap on every write operation to the data unit. For example, when we need to write a narrow-width value into the cache, the swap status of the current data unit will be checked. If the current data in the cache is in a non-swapped status, we will swap the lower half and the upper half of the incoming narrow-width value and only store the lower half of the narrow width value into the upper half of the data unit in the cacheline. If the current data in the cache is in a swapped status, normal leading zeros masking write will be performed and only the lower half of the incoming value will be written into the lower half of the data unit in the cacheline.

2) Swap on Replacement: Since the swap operation for the SW scheme is based on the write operation to the cache, the overhead especially the energy overhead for swap will be still high. In order to further reduce the overhead of the swap operation, we propose the swap on replacement (SRepl) scheme, where the swap is done on every replacement of the cacheline. During the cacheline replacement, if a narrow-width value needs to be written into the cacheline, we will check the swap status of the current data unit and do the similar swap to that in the SW scheme. Since the miss rate of the cache is usually very low, the overall overhead of the SRepl scheme will be much lower than the SW scheme. However, due to the reduced swap frequency of the SRepl scheme, the effectiveness of the write variations balancing in SRepl scheme will also be reduced compared to the SW scheme.

C. Microarchitecture of the Non-Volatile Cache Design

To implement our leading zeros masking and swap design, one additional narrow bit (N) for each data unit (64-bit) and one additional swap bit (S) for each cacheline are needed. The narrow bit (N) indicates whether the value in the current data unit is narrow-width or not. The swap bit (S) indicates whether the data in the current cacheline is in the swapped or non-swapped status. Note that since the access to last level cache is cacheline-based, we choose the cacheline-level swap bit stead of one S bit per 64-bit unit to further reduce the space overhead of our design. For instance, for a last level cache with a 128-byte cacheline, the space overhead of our non-volatile cache compared to the data array is only (1/64 + $1/(128\times8)$ = 1.7%).

Figure 2 shows the block diagram of our proposed non-volatile cache design with leading zeros masking and swap. In the tag array, one narrow bit (N) is added for each 64-bit data unit in the data array, and one swap bit (S) is added for each cacheline. During a cache write, the narrowness of the

incoming data will be checked first. If the data is not narrow-width, a normal write will be performed and the N bit will be set to zero. If the data is a narrow-width data, the S bit will be checked, and the swap will be done for the swap on write (SW) scheme. Otherwise, the swap will be only done during cache write caused by the cache replacement for the swap on replacement (SRepl) scheme. Based on the status of the S bit, the lower half of the narrow width value will be written into the lower or the upper half of the data unit as discussed above. During a cache read, the N bit will be checked first. If the N bit is zero indicating that the data is not narrow-width, the data will be readout in its current format. If the N bit is one, the S bit will be also checked. If the S bit is zero, it means that the narrow width data is currently in the non-swapped status. We can read out the lower half of the data and do the zero-extension to restore the original 64-bit value. If the S bit is one, we need to readout the upper half of the data and do the zero-extension.

D. Optimization Schemes

To further improve the lifetime our proposed non-volatile cache, we propose to adopt two optimization schemes to reduce the write stress on the cacheline.

1) Multiple Dirty Bit: The multiple dirty bit (MDB) scheme has been proposed for improving the reliability of the data cache against soft errors [15]. The idea is that not all the words in a dirty cacheline are modified (dirty) during the write-back. Therefore, if we can have one dirty bit for each word, we can only write back the dirty words into the lower cache to avoid unnecessary writes for the clean words and also save the energy. We adopt this MDB scheme for reducing the write stress to the last level cache and thus improving its lifetime.

2) Read Before Write: The read before write (RBW) scheme was previously proposed for reducing the amount of writes in the PCM memory design [1][6]. A pre-read operation is performed before the write to eliminate the write of the new value same to its previous value. By combining our proposed MDB scheme, we can apply a selective RBW by only pre-read the data unit that the dirty words of upper level cache will be written into, to further reduce the overhead caused by the pre-read operation.

IV. Experimental Evaluation

A. Experimental Setup

We derive our simulator from SimpleScalar V3.0 [16] to model a high-performance microprocessor similar to Alpha 21364. Table I gives the detailed configuration of the simulated microprocessor. For experimental evaluation, we use the SPEC CPU2006 benchmark suite compiled for the Alpha Instruction Set Architecture (ISA). Ten benchmarks are randomly selected for our experimental evaluation. We use the reference input sets for this study. We first fast-forward 500 million instructions and then simulated the next 500 million instruction in detail.

B. Lifetime Metrics

Since the lifetime of the no-volatile memory cell (bit) is depending on the amount of writes to the cell, we define our Lifetime Improvement (LI) for a data unit as follows:

TABLE I
Parameters of the simulated processor.

Processor Core	
Datapath Width	4 inst. per cycle
Int Issue Queue	20 entries
FP Issue Queue	15 entries
Load/Store Queue	64 entries
Active list (ACL)	80 entries
Int Register File	80 registers
FP Register File	72 registers
Function Units	4 IALU, 2 IMULT/IDIV 2 FALU, 1 FMULT/FDIV/FSQRT 2 MemPorts
Branch Predictor	
Branch Predictor	Alpha 21264 tournament predictor 32-entry RAS
BTB	2048-entry 2-way
Memory Hierarchy	
L1 I/DCache	64KB, 2 ways, 64B blocks, 2 cycles
L2 UCache	4MB, 8 ways, 128B blocks, 12 cycles
Memory	225 cycles first chunk, 12 cycles rest
TLB	Fully-assoc., 128 entries

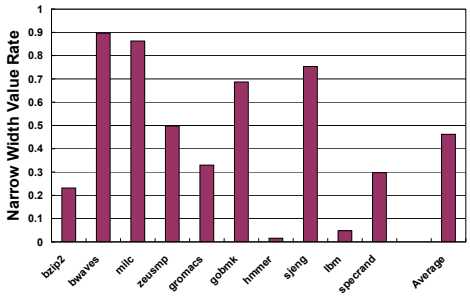


Fig. 3. The percentage of the narrow-width values (leading zeros) in the last level cache.

$$LI = MIN(\frac{W_{upper\_org} - W_{upper\_impr}}{W_{upper\_impr}}, \frac{W_{lower\_org} - W_{lower\_impr}}{W_{lower\_impr}}) \quad (1)$$

where W_{upper_org} is the number of bits needed to be written into the upper half in the original cache design and W_{lower_org} is the number of bits needed to be written into the lower half in the original cache design. W_{upper_impr} is the number of bits needed to be written into the upper half in our lifetime improvement scheme and W_{lower_impr} is the number of bits needed to be written into the lower half in our lifetime improvement scheme. Since the lifetime of the cache is depending on the first cell that fails. Therefore, we define our lifetime improvement for the data unit as the minimum of lifetime improvement for the upper half and the lower half.

C. Experimental Results and Analysis

In order to study our leading zeros masking scheme, we first conduct the experimental analysis on the percentage of the narrow-width values in our simulated last level cache. As we mentioned above, the data unit we choose is 64-bit. Figure 3 shows that on average 46.2% of the values in the last level cache is narrow-width, which means that 46.2% of the 64-bit data in the cache have all zeros in their leading 32 bits. Therefore, our leading zeros masking scheme has very high possibility in reducing the write stress in the upper half of the data unit.

Only applying the leading zeros masking scheme will not reduce the amount of write operations to the lower half of the data unit. Therefore, we proposed two swap schemes, the

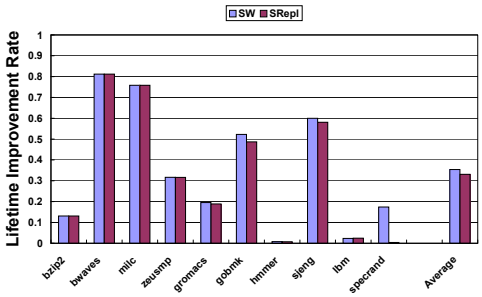


Fig. 4. The lifetime improvement rate by applying the swap on write (SW) and swap on replacement (SRepl) schemes.

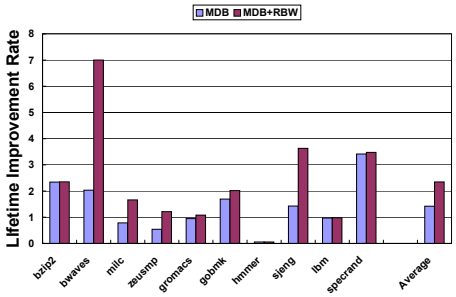


Fig. 5. The lifetime improvement rate of the multiple dirty bit (MDB) and read before write (RBW) schemes.

swap on write (SW) and the swap on replacement (SRepl), to further balance the write variations between the lower and upper halves. We first apply the SW scheme to balance the write variation and Figure 4 shows that the lifetime of our non-volatile cache will be improved by 35.4% on average. Note that the improvement rates of the *hmmer* and *lbm* are relatively low because the percentage of the narrow-width values in these two applications are very low as shown in Figure 3. Therefore, the effectiveness of our lifetime improvement scheme will be hurt.

Figure 4 also shows that if we apply the swap on replacement (SRepl) scheme, the lifetime improvement rate will be reduced to 33.1%, which is slightly less than the SW scheme. However, as we discussed in Section III, the overhead of the SRepl scheme will be lower than the SW scheme. Therefore, we choose the SRepl scheme as the default scheme for our following study.

In order to further improve the lifetime of our non-volatile cache, we proposed to adopt two optimization schemes, the multiple dirty bit (MDB) and read before write (RBW) schemes. For the space overhead consideration, we choose a 64-bit level dirty bit for the upper level cache, i.e., the data cache in our study. For the RBW scheme, we also consider the 64-bit level control for selective pre-read combined with the MDB scheme, which means that for a write-back dirty word (64-bit), if the word is the same as the old word in the cache, the write operation will be avoided. If two values are different, the dirty word will be written into the cache according to the proposed leading zeros masking and swap schemes. Figure 5 shows that if we only adopt the MDB scheme, the lifetime improvement rate will be improved to 142% on average. If we combine the MDB and RBW schemes, the lifetime improvement rate will further increase to 245% on average.

V. Conclusion and Future Work

In this paper, we propose to exploit the narrow-width values to improve the lifetime of the non-volatile last level caches. We first propose a leading zeros masking (LZM) scheme to reduce the amount of write operations to the upper half of the narrow-width data. Then, two swap schemes, the swap on write (SW) and swap on replacement (SRepl), are proposed to mitigate the write variations between the upper half and the lower half of the narrow-width data. To further reduce the write stress to the non-volatile cache, we propose to adopt two optimization schemes, the multiple dirty bit (MDB) and read before write (RBW). The experimental results show that our proposed schemes can improve the lifetime of the non-volatile caches by 245% on average. In future work, we will study the write variation between cachelines and evaluate the energy consumption reduction in our proposed schemes.

Acknowledgment

This work was supported in part by a grant from Chinese NSF Award 61100035.

References

[1] P. Zhou *et al.*, "A durable and energy efficient main memory using phase change memory technology," in *Proc. of the Int'l Symposium on Computer Architecture*, 2009, pp. 14–23.

[2] M. K. Qureshi *et al.*, "Enhancing lifetime and security of pcm-based main memory with start-gap wear leveling," in *Proceedings of IEEE/ACM International Symposium on Microarchitecture*, 2009.

[3] N. H. Seong *et al.*, "Security refresh: prevent malicious wearout and increase durability for phase-change memory with dynamically randomized address mapping," in *Proc. of the Int'l Symposium on Computer Architecture*, 2010.

[4] R. Rodriguez-Rodriguez *et al.*, "Reducing writes in phase-change memory environments by using efficient cache replacement policies," in *Proceedings of the Conference on Design, Automation and Test in Europe*, 2013.

[5] S. Cho and H. Lee, "Flip-n-write: a simple deterministic technique to improve pram write performance, energy and endurance," in *Proceedings of IEEE/ACM International Symposium on Microarchitecture*, 2009.

[6] Y. Joo *et al.*, "Energy- and endurance-aware design of phase change memory caches," in *Proc. of the Design, Automation and Test in Europe Conference (DATE)*, 2010, pp. 136–141.

[7] J. Wang *et al.*, "i2wap: Improving non-volatile cache lifetime by reducing inter- and intra-set write variations," in *Proceedings of the International Symposium on High Performance Computer Architecture*, 2013.

[8] S. Schechter *et al.*, "Use ecp, not ecc, for hard failures in resistive memories," in *Proceedings of the International Symposium on High Performance Computer Architecture*, 2010.

[9] E. Ipek *et al.*, "Dynamically replicated memory: building reliable systems from nanoscale resistive memories," in *Proceedings of the International Symposium on Architectural Support for Programming Languages and Operating Systems*, 2010.

[10] D. H. Yoon *et al.*, "Free-p: Protecting non-volatile memory against both hard and soft errors," in *Proceedings of the International Symposium on High Performance Computer Architecture*, 2011.

[11] D. Brooks and M. Martonosi, "Dynamically exploiting narrow width operands to improve processor power and performance," in *Proc. of HPCA-5*, 1999.

[12] O. Ergin *et al.*, "Exploiting narrow values for soft error tolerance," *IEEE Computer Architecture Letters*, vol. 5, no. 2, July-Dec. 2006.

[13] J. Kong and S. W. Chung, "Exploiting narrow-width values for process variationtolerant 3-d microprocessors," in *Proc. of the 49th Design Automation Conference (DAC)*, 2012, pp. 1193–1202.

[14] L. Villa, M. Zhang, and K. Asanovic, "Dynamic zero compression for cache energy reduction," in *Proc. of Micro-33*, 2000, pp. 214–220.

[15] S. Wang, J. Hu, and S. G. Ziavras, "On the characterization and optimization of on-chip cache reliability against soft errors," *IEEE Transactions on Computers*, vol. 58, no. 9, pp. 1171–1184, September 2009.

[16] D. Burger and T. M. Austin, "The simplescalar tool set, version 2.0," Computer Sciences Department, University of Wisconsin, Tech. Rep. 1342, 1997.