Adaptive replacement policy for hybrid cache architecture

Ju-Hee Choi\textsuperscript{1a)} and Gi-Ho Park\textsuperscript{2b)}

1 Department of Computer Science and Engineering, Seoul National University, Gwanak-ro, Gwanak-gu, Seoul, 151–742, Republic of Korea
2 Department of Computer Engineering, Sejong University, 98, Gunja-Dong, Gwangjin-Gu, Seoul, 143–747, Republic of Korea

\textsuperscript{a)} loader77@snu.ac.kr
\textsuperscript{b)} ghpark@sejong.ac.kr, corresponding author

Abstract: Researchers have proposed several schemes with hybrid cache architecture (HCA) which contains both SRAM cells and non-volatile memory (NVM) cells to overcome the drawbacks of NVM. The existing HCA schemes try to place a write intensive block into an SRAM way when a cache miss occurs. The cache replacement policy increases the write counts of NVM when the number of write intensive blocks are small. To solve this problem, we propose an adaptive replacement policy for hybrid cache architecture to adjust the cache replacement policy based on the write intensity for victim selection. The simulation results show that the proposed mechanism reduces the write counts of NVM by 21.2\% on average compared that conventional replacement policy is used.

Keywords: non volatile memory, STT-RAM, cache replacement policy, low power

Classification: Integrated circuits

References

[1] S. Raoux, G. W. Burr, M. J. Breitwisch, C. T. Rettner, Y. C. Chen, R. M. Shelby, M. Salinga, D. Krebs, S. H. Chen and H. L. Lung: IBM J. Res. Develop. \textbf{52} (2008) 465. DOI:10.1147/rd.524.0465
[2] P. Zhou, B. Zhao, J. Yang and Y. Zhang: ICCAD 2009 (2009) 264. DOI:10.1145/1687399.1687448
[3] H. Kim, J. H. Ahn and J. Kim: IEEE Trans. Very Large Scale Integr. (VLSI) Syst. \textbf{21} (2013) 1863. DOI:10.1109/TVLSI.2012.2220791
[4] X. Wu, J. Li, L. Zhang, E. Speight and Y. Xie: DATE 09 (2009) 737. DOI:10.1109/DATA.2009.5090762
[5] J. Li, L. Shi, C. J. Xue, C. Yang and Y. Xu: ESTIMedia11 (2011) 19. DOI:10.1109/ESTIMedia.2011.6088521
[6] J. Ahn, S. Yoo and K. Choi: ISLPED 2013 (2013) 223. DOI:10.1109/ISLPED.2013.6629298
[7] Georgia Institute of Technology: \url{http://comparch.gatech.edu/hparch/macsim/macsim.pdf} (2012).
[8] J. L. Henning: ACM COMP. AR. \textbf{34} (2006) 1. DOI:10.1145/1186736.1186737
1 Introduction

Non-volatile memory (NVM) such as phase change memory (PCM) [1] and spin-transfer torque RAM (STT-RAM) [2] is a promising technology for its extremely low leakage power. The static energy consumption of last level cache (LLC) occupies more than 30% of the total energy consumption, thus, employing NVM as LLC largely decreases the energy consumption of the core [3]. However, the write operation of NVM should be carefully controlled because it needs longer latency, higher power consumption and shorter lifetime compared to SRAM.

Hybrid cache architecture (HCA) has been proposed to decrease the number of write operations of NVM [4, 5, 6]. HCA is basically composed of NVM, but it also has a small amount of SRAM to reduce write pressure on NVM. The role of SRAM in HCA has been to accommodate the write intensive blocks. Therefore, the existing HCA schemes focus on how to detect the write intensive blocks and how to place them into SRAM to decrease the number of write access to NVM.

Previous works commonly store an incoming block into an SRAM way or an NVM way according to write intensity of the block when a cache miss occurs. If the block is a write intensive block, a victim block is selected only in SRAM ways. Otherwise, the victim block is chosen in NVM ways. We define this cache replacement policy as write intensity dependent replacement policy (WIDRP). On the contrary, write intensity independent replacement policy (WIIRP) means that write intensity of an incoming block is not considered during victim selection. Most of existing HCA schemes can be categorized into WIDRP. However, this approach is not suitable for all applications.

If the rate of the write intensive blocks over the total blocks is low, WIIRP is more beneficial than WIDRP for reducing the number of write accesses to NVM. In WIDRP, blocks of SRAM ways rarely participate in victim selection and most of victim blocks are selected in only NVM ways for an application that has a small number of the write intensive blocks. In that case, blocks of NVM ways in WIDRP are replaced by fetched blocks more frequently than in WIIRP, which leads to increase the write requests to NVM. If the cache replacement policy is adaptively switched to a more efficient policy according to the rate of the write intensive blocks, the write counts of NVM is decreased compared to that when only one policy is used.

In this paper, we introduce a new hybrid cache architecture to decrease the write operations of NVM. In our scheme, the rate of the write intensive blocks over the total blocks is low, WIIRP is more beneficial than WIDRP for reducing the number of write accesses to NVM. In WIDRP, blocks of SRAM ways rarely participate in victim selection and most of victim blocks are selected in only NVM ways for an application that has a small number of the write intensive blocks. In that case, blocks of NVM ways in WIDRP are replaced by fetched blocks more frequently than in WIIRP, which leads to increase the write requests to NVM. If the cache replacement policy is adaptively switched to a more efficient policy according to the rate of the write intensive blocks, the write counts of NVM is decreased compared to that when only one policy is used.

In this paper, we introduce a new hybrid cache architecture to decrease the write operations of NVM. In our scheme, the rate of the write intensive blocks is monitored and a cache controller is able to select WIDRP or WIIRP adaptively. If the rate falls below a threshold, WIIRP is used to select a victim block. Otherwise, WIDRP is chosen for victim selection. Simulation results show that our method reduces the number of write access to NVM by 21.2% on average compared that of the conventional replacement policy.

2 Analysis of existing HCA schemes

To analyze the impact of the two cache replacement policies, we examined the normalized write counts of NVM of read write aware hybrid cache architecture (RWHCA) [4] and write intensity prediction (WIP) [6] with both policies, WIDRP
and WIIRP. We also investigated the rate of the write intensive blocks which is calculated by dividing the number of the write intensive blocks over the total number of blocks. Fig. 1 shows these two metrics and all programs are sorted based on the rate of the write intensive blocks as increasing order from left to right.

Overall, WIDRP is more efficient than WIIRP as write counts of NVM is reduced by 17.3% in RWHCA and 22.0% in WIP with WIDRP compared to each scheme with WIIRP. However, the normalized values of the programs vary with the rate of the write intensive blocks. For HCA schemes with WIDRP, the normalized number of write accesses to NVM exceeds one in almost all of programs on the left half side which have low rate, while most of programs on the right half side show a decrease in the number of write accesses to NVM. It means that WIIRP is more efficient than WIDRP when the rate is less than 20%.

To investigate this phenomenon, we examined the impact of the rate of the write intensive blocks per set on the NVM write counts because no set has the same rate. As some researchers have appointed [5], write intensity varies across sets, thus, we check the relation between the change of write counts of NVM and the rate of the write intensive blocks for each set. Fig. 2 shows the difference of write

Fig. 1. Normalized write counts of NVM and write intensive block rate of RWHCA (a) and WIP (b) with WIIRP and WIDRP. WIIRP is the standard of normalization.

Fig. 2. Write counts of NVM difference per set between WIIRP and WIDRP according to write intensive block rate. $\text{Diff} = \text{NVM write counts in WIIRP} - \text{NVM write counts in WIDRP}$. 
counts of NVM and the rate per set across all applications. Overall, if the rate is less than 20%, the write counts of NVM in WIDRP is larger than that in WIIRP. On the contrary, if the rate is more than 20%, the number of write access to NVM in WIDRP is less than that in WIIRP in almost of all sets. Therefore, we found that any single policy cannot efficiently reduce the write counts of NVM for all sets.

Based on these observations, we propose a cache replacement policy to select an appropriate replacement policy for each set by monitoring the rate of the write intensive blocks for each set during the execution. The following chapter shows a detailed description of our proposed scheme.

3 Adaptive replacement policy for hybrid cache architecture

We propose a novel scheme which is called adaptive replacement policy for hybrid cache architecture (ARPHCA) to decrease the number of write access to NVM as shown in Fig. 3. In our proposal, we adaptively choose either one of WIDRP or WIIRP according to the rate of the write intensive blocks during line fill operation. In addition, the cache replacement policy can be applied for each set of LLC because no set has the same rate of the write intensive blocks.

To implement our proposed scheme, we introduce write intensive block monitor (WIBM) whose role is to keep track of the information for calculating the rate dynamically. An entry of WIBM consists of a total block counter (TBC) and a write intensive block counter (WIBC). Each entry has its own corresponding set of LLC; thus, the number of WIBM entries is the same as that of the LLC set. The TBC indicates the number of total blocks and a WIBC does the number of the write intensive blocks. If the rate of the write intensive block is smaller than a certain threshold, WIIRP is used as a replacement policy for that set, applied when a cache miss occurs. Otherwise, WIDRP is applied. We use 20% as the threshold in this paper. The proposed mechanism decreases the values of the TBC and the WIBC when the TBC is saturated.

Fig. 4 presents the operational flow for the proposed mechanism. First, whenever a block is detected as a write intensive block, a WIBC of the corresponding set is increased by 1 (lines 1–3). When a cache miss occurs, the corresponding TBC is

| Algorithm: Choosing Victim Block |
|----------------------------------|
| **parameters:** |
| TBC: Total Block Counter |
| WIBC: Write Intensive Block Counter |
| SN: Set Number |
| WIIRP: Write Intensive Block Rate |
| **procedure:** |
| 1: If a write intensive block is detected then |
| 2: WIBC[SN] ← WIBC[SN] + 1 |
| 3: else |
| 4: If a cache miss occurs then |
| 5: TBC[SN] ← TBC[SN] + 1 |
| 6: Calculate WIIRP Rate |
| 7: If WIIRP Rate < Threshold then |
| 8: Select Victim Block using WIIRP |
| 9: end |
| 10: Select Victim Block using WIDRP |
| 11: else |
| 12: If TBC[SN] is saturated then |
| 13: TBC[SN] ← TBC[SN] / 2 |
| 14: WIBC[SN] ← WIBC[SN] / 2 |
| 15: end |
| 16: end |

Fig. 3. Structure of ARPHCA. Fig. 4. Algorithm for choosing victim block.
increased by 1 (lines 4–5). Next, the rate of the write intensive blocks is calculated and compared with the threshold rate (line 6–7). If it is lower than the threshold, a victim is selected regardless of write intensity. It means that the WIIRP is used for victim selection (line 8). Otherwise, a new cache block is placed either an SRAM way or an NVM way according to write intensity of the incoming block (lines 9–10). When the TBC becomes saturated, the values of the corresponding TBC and WIBC are halved (lines 12–15).

4 Experimental setup and results

4.1 Experimental setup

The simulation was performed using Macsim [7] which is a trace-driven and cycle level simulator. It is designed to thoroughly model the detailed micro-architectural behavior, including pipeline stages and memory systems. Our baseline system has a two-level cache hierarchy. The L1 cache is composed of SRAM memory, while the LLC is a hybrid cache system which consists of SRAM and STT-RAM memories. We set the base configuration as proposed in [6]. Table I shows our baseline processor configurations in detail. We also used SPEC CPU2006 benchmark suite [8]. ARPHCA is designed to use WIP as a base mechanism which is the most effective HCA scheme. In addition, ARPHCA dynamically chooses between WIDRP and WIIRP. Since each TBC and WIBC has 10 bits, the total storage overhead is less than 1% (2048 * 20 bits = 5 KB, 5 KB/2 MB = 0.002441).

4.2 Simulation results

We examined the ratio of choosing WIIRP during linefill operation in ARPHCA as shown in Fig. 5. As we expected, WIIRP is more frequently chosen than WIDRP for the applications which have low rate of the write intensive blocks during victim selection. On the contrary, right half applications in Fig. 5 chose WIDRP more frequently than WIIRP. The figure also depicts the accuracy of selection, which means the rate of the selections which benefit by reducing the write requests of NVM in effect. We achieved more than 80% accuracy over three fourths of all programs, while some programs such as wrf and gcc have low accuracy.

| Table I. Processor configurations |
|-----------------------------------|
| Core Type                         | x86, out-of-order, 2 GHz           |
| INT/MEM/FP                        | 4/4/4                              |
| Branch Predictor                  | gshare predictor, 16 history length|
| ROB Size                          | 256                                |
| I-Cache/D-cache                   | 16 KB, 4-way, 64 B blocks, 1-cycle latency |
| Last Level Cache                  | 2 MB (12-way STT-RAM and 4-way SRAM), 64 B blocks STT-RAM: 9-cycle (read) and 28-cycle (write) latency SRAM: 8-cycle latency |
| Memory Latency                    | 200 cycles                         |
| TBC/WIBC                          | 10 bits/10 bits                    |
| WIBM Entries                      | 2048                               |

© IEICE 2014
DOI: 10.1587/elex.11.20140946
Received October 1, 2014
Accepted October 17, 2014
Published November 4, 2014
Copyedited November 25, 2014
Fig. 6 shows the normalized number of write access to NVM of WIP with WIDRP, WIP with WIIRP, and ARPHCA. All left side applications show nearly the same normalized value of WIP with WIIRP. In addition, the normalized write counts of ARPHCA are close to those of WIP with WIDRP for almost all of right side programs in Fig. 6. These results are consistent with the ratio of choosing WIIRP and the accuracy of selection. When the ratio and the accuracy are both more than 80%, the normalized values of ARPHCA follow the trend of WIIRP. On the other hand, if the ratio is less than 10% and the accuracy is over 80%, the normalized values of ARPHCA are similar to those of WIDRP. Overall, we achieved the reduction of average write counts of NVM in ARPHCA by 21.2% and 93.6% at maximum compared to WIP. In summary, ARPHCA usually selects the appropriate replacement mechanism and can achieve much more write count reduction than any one of WIIRP or WIDRP mechanisms.

5 Conclusion and future work

This paper proposed a new mechanism called adaptive replacement policy for hybrid cache architecture (ARPHCA). To gain new insights on the HCA scheme, we analyzed the characteristics of the previously suggested HCA schemes. The analysis showed that WIDRP had negative effects on the applications which have the low rate of the write intensive blocks. Based on the analysis, our scheme dynamically selects the cache replacement policy according to the rate of the write intensive blocks. The simulation result showed that the proposed mechanism
achieved 21.2% reduction in the number of write access to NVM compared to WIP on average.

It is expected that adopting a wear leveling (WL) scheme can achieve more write access reduction with our proposal. WL makes the write operations uniform by switching write intensive cells to less frequently written cells. However, simply applying WL schemes with our scheme would not be a good solution. To take advantages of WL, ARPHCA needs to keep track of the number of write accesses to all NVM cells, which causes a large area overhead and lots of extra energy consumption for monitoring. Thus, devising a new HCA mechanism to apply WL with our proposal would be a promising future research.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (NRF-2011-220-D00098).