IEICE TRANS. INF. & SYST., VOL.E97–D, NO.8 AUGUST 2014

LETTER

Write Avoidance Cache Coherence Protocol for Non-volatile Memory as Last-Level Cache in Chip-Multiprocessor*

Ju Hee CHOI†, **Jong Wook KWAK**††a), *Nonmembers*, *and* **Chu Shik JHON**†, *Member*

SUMMARY Non-Volatile Memories (NVMs) are considered as promising memory technologies for Last-Level Cache (LLC) due to their low leakage and high density. However, NVMs have some drawbacks such as high dynamic energy in modifying NVM cells, long latency for write operation, and limited write endurance. A number of approaches have been proposed to overcome these drawbacks. But very little attention is paid to consider the cache coherency issue. In this letter, we suggest a new cache coherence protocol to reduce the write operations of the LLC. In our protocol, the block data of the LLC is updated only if the cache block is written-back from a private cache, which leads to avoiding useless write operations in the LLC. The simulation results show that our protocol provides 27.1% energy savings and 26.3% lifetime improvements in STT-RAM at maximum.

key words: *cache coherence, non volatile memory, STT-RAM, chip multiprocessor*

1. Introduction

Researches on non-volatile memories (NVMs), such as STT-RAM or PCM, have substantially increased in recent years [1], [2]. Unlike the conventional memories, NVMs consume little or no static power because they do not use the electric power to store their information. Therefore, the static energy consumption can be remarkably reduced, if the cache is composed of NVMs. Since the leakage power is responsible for a significant part of the power consumption in the last-level cache (LLC), researchers have studied to use NVMs as an alternative to SRAM in the LLC [3], [4].

Although NVMs are very attractive alternative for SRAM, there are three penalties to use. Modifying the states of NVM cells requires long latency and high level current. In addition, NVM cells have limited endurance. To overcome these difficulties, several methods were suggested [3], [4]. However, we found that the previous works have not considered the cache coherence issue. Under the environments of SRAM-based LLC, these drawbacks of the write operation of NVMs are not considered during designing the cache coherence protocol. Therefore, the existing cache coherence protocols [5] have a potential to reduce the write operations, which leads to decreasing the dynamic energy consumption and enhancing the write endurance. In this letter, we introduce a new cache coherence protocol for NVMs to decrease the number of write access to the LLC. In our scheme, the data array of the LLC is not updated during the linefill operation, while the tag array is changed to maintain the inclusion property. The data array is modified only when the cache block is written-back from the private cache. Our protocol reduces the number of write access to the LLC; thus, the dynamic energy consumption is reduced and the lifetime is enhanced in our protocol.

In this letter, we introduce a new cache coherence protocol for NVMs to decrease the number of write access to the LLC. In our scheme, the data array of the LLC is not updated during the linefill operation, while the tag array is changed to maintain the inclusion property. The data array is modified only when the cache block is written-back from the private cache. Our protocol reduces the number of write access to the LLC; thus, the dynamic energy consumption is reduced and the lifetime is enhanced in our protocol.

2. Motivation

In this section, we review the legacy cache coherence protocols to get a new insight to reduce the write operations. The existing studies about cache coherence protocol have not concentrated on reducing the write operations because it does not matter in the SRAM-based LLC. Since there is no drawback of write operation compared to read operation, the number of write access is not taken into account. However, reducing the write operations is an important issue in NVM-based LLC. The dynamic energy consumption is largely depend on the write operations, because the dynamic energy of write operation is greater than that of read operation. Moreover, the lifetime is inversely proportional to the number of write access. Therefore, a new protocol for NVMs to minimize the write operations is needed.

There are useless write operations in the existing protocol. Generally, memory systems of CMPs are composed of a shared LLC and several private caches which are dedicated to cores [5]. In addition, the cache block is divided into two arrays: tag array and data array. Tag array stores tag bits and cache coherence state, while data array stores block data. When a linefill operation occurs at the LLC, the requested block data is written to the data array, and the tag bits and cache coherence state are updated to the tag array. Then, the cache block is forwarded and linefilled to the private cache. When a core tries to modify the cache block in the private cache, an invalidation signal is sent to the shared

Manuscript received March 3, 2014.

†The authors are with the Department of Computer Science and Engineering, Seoul National University, Korea.

††The author is with the Department of Computer Engineering, Yeungnam University, Korea.

*This work has been funded by the BK21+ program of the National Research Foundation (NRF) of Korea.

a) E-mail: kwak@ynu.ac.kr

DOI: 10.1587/transinf.E97.D.2166

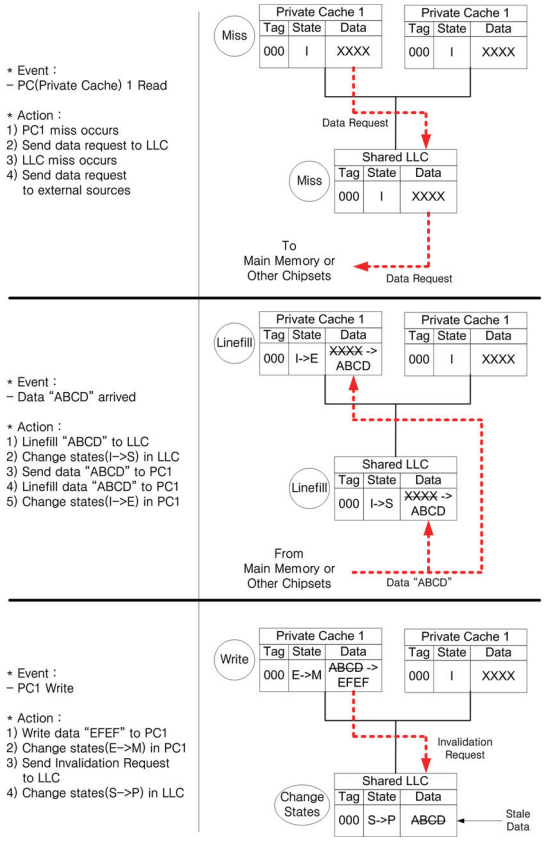

Fig. 1 Conventional protocol.

Table 1 States and descriptions.

State	Description
I(nvalid)	The cache block is invalid
S(hared)	The cache block has valid block data and other private caches may have valid copy.
E(xclusive)	The cache block has valid block data with exclusive permission and other caches have no valid copy.
M(odified)	The cache block has valid and modified block data. Other caches have no valid copy. This state appears in the private cache only.
P(rivate cache)	The cache block in the LLC has no valid block data, but more than one of the private caches has valid block data. This state appears in the LLC only.

* P state is introduced due to keeping the inclusion property. Modern multiprocessors have employed the inclusive LLC to filter the cache coherence traffic from other chipset or the main memory. Thus, it is needed that a state represents one of the private caches has valid data even the LLC has no valid data.

LLC and other private caches to maintain the cache coherence. Therefore, the previous write access to the LLC during the linefill operation is considered as the useless write operation, if the cache block in the LLC has been never used until it is invalidated.

Figure 1 illustrates an example of write inefficiency in widely used cache coherence protocols such as MESI or MOESI [6]. In the example, we assume that a core reads and writes a block data of the PC (Private Cache) 1. Table 1 lists the cache states in the figure and their descriptions. When the core tries to read the block data, since the PC1 has no valid block data, the cache controller sends the request for the block data to the LLC.

However, the LLC also has no valid copy; thus, the request is sent to the external sources such as the main memory or other chipsets. When the block data "ABCD" is arrived at the LLC, it is written into the LLC and the state of the LLC is changed to S state, which means the cache block is valid and other private caches may have the same cache block. Then, the block data "ABCD" is forwarded to the PC1.

When the block data is received in the PC1, it is written into the PC1 and the state of the PC1 is changed to E state. After the linefill operation is completed, if the core tries to modify the block data "ABCD" to "EFEF", an invalidation request is sent to the LLC to maintain cache coherence. The purpose of the invalidation request is indicating that the block data of the PC1 is modified and the cache block in the LLC should be invalidated. If the block data "ABCD" in the LLC has not been used until it is invalidated, writing the block data "ABCD" to the LLC during the linefill operation was an useless write operation.

3. Write Avoidance Cache Coherence Protocol

To deal with this problem, we suggest a new cache coherence protocol which is called Write Avoidance Cache Coherence (WACC) protocol. In our protocol, the block data of the cache block is not written into the LLC during the linefill operation, while the tag bits and the cache coherence state are updated. Since the block data is not placed in the LLC, one of the private caches has responsibility to provide the valid block data. The block data in the LLC is only updated when it is written-back from the private cache. The writeback is initiated only when no other private cache has the block data in WACC protocol. Therefore, we avoids useless write operation due to modifications of the block data in the private cache.

Figure 2 shows an example of WACC protocol. Unlike the conventional protocols, when the block data "ABCD" is arrived at the LLC, it is not written to the LLC. Instead, the state is changed to P state and the block data is forwarded to the PC1. When the PC1 is modified to "EFEF", there is no need to send an invalidation request to the LLC for the block data "ABCD" is not written to the LLC. Therefore, one write operation of the LLC and one request for cache coherence is decreased compared to the baseline protocols in this case.

We compare an simple version of the existing MOESI protocol with its modified protocol in Fig. 3. Table 2 shows the coherence signals and actions. The transition signal is divided into two parts: {signal}_{source} and the action indicates the operation of the data array. For example, WB_PC/Wr means that if the block is P state and receives the WB signal from a private cache, the block data is written to the data array.

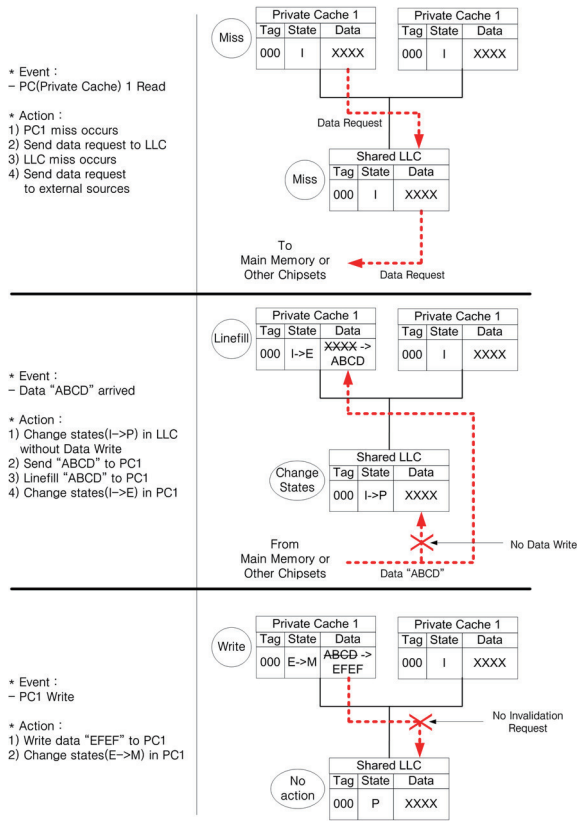

Fig. 2 Write avoidance cache coherence protocol.

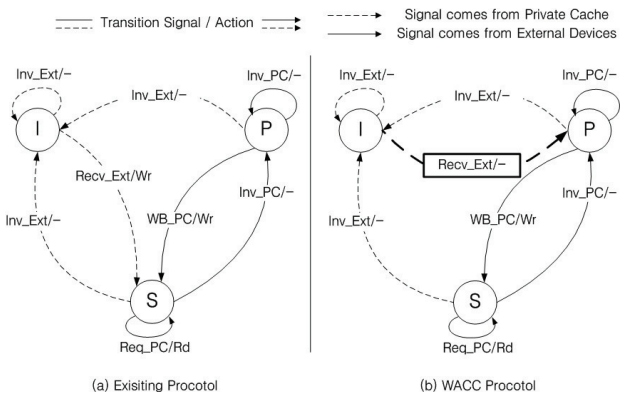

Fig. 3 State transition diagrams for LLC. Bold lines indicate modified transition signals and actions.

As shown in Fig. 3 (a), when a new cache block is received in the LLC, the state of the cache is transition to S state and the block data is written to the data array in the existing protocol. On the contrary, the state is transition to P state instead of S state in our protocol under the same condition. Furthermore, the write operation is omitted as shown in Fig. 3 (b). This is because the block data is forwarded without write access to the data array in WACC protocol.

Another point to be considered is that the protocol of the private cache should be changed. The writeback operation is initiated if the cache block in the private cache is modified and evicted in the existing protocols. However, the cache block should be written-back to the LLC in WACC protocol when it is evicted in the private cache regardless of whether the cache block is dirty or not.

Table 2 Signals/actions and descriptions.

Signal	Description
`Inv`	Invalidate the cache block if it is valid. This signal is generated when another device tries to modify the block data.
`Recv`	Provide the block data in the cache block. This signal is generated when a cache hit occurs.
`Req`	Request the block data for read operation. This signal is generated when a cache miss occurs.
`WB`	Writeback the block data to the LLC. This signal is generated when a private cache evicts the cache block.
Action	**Description**
`Wr`	Write the block data of the received cache block into the data array.
`Rd`	Read the block data and provide it with the requestor.

4. Performance Evaluation

We simulated our approach with PARSEC benchmark suite [7]. The *gem5* simulator is used to evaluate the normalized energy and normalized lifetime of our protocol [8]. The overall simulation parameters are shown in Table 3. We assume that the cache coherence protocol is a MOESI protocol. In addition, LLC is composed of STT-RAM because STT-RAM is considered as the right alternative among several NVMs [3]. The power value of STT-RAM is derived from the previous work [9].

Figure 4 presents the normalized number of the read and write access to LLC in our protocol compared to the baseline MOESI protocol. Note that write access is divided into writeback access and linefill access. As a result, 13.2% of the write operations were decreased on average. The noticeable result is that the number of the writeback access was increased, while there were no linefill operation. When a cache block is evicted in a private cache, the writeback operation is not required in the existing protocols if the cache block is not modified. This is because the LLC already has the valid block data if the cache block is clean. On the contrary, the writeback operation should be initiated if no other private cache has the valid copy during cache replacement in WACC protocol. This difference generates the extra writeback operations. However, the total number of the write access in WACC protocol is smaller than that of other protocols because the reduction in the linefill operation is much larger than the increment in the writeback operation.

We show the normalized dynamic energy consumption and lifetime in Fig. 5. Since the dynamic energy in write operation dominates the dynamic energy consumption in read operation, the reduction of the write operations leads to reducing the total dynamic energy consumption. Our protocol achieves 27.1% energy savings at maximum and 10.8% energy savings on average. In addition, WACC protocol also extends the lifetime of the LLC because the lifetime of STT-RAM is inversely proportional to the number of write access

Table 3 Parameters of the simulated architecture.

Parameter	Value
Cores	4
L1 Inst / Data Cache	64KB, 2-way, 64B line
L2 Unified Cache	2MB, 16-way, 64B line
Memory	64bit bus width, 4 read/write ports
Function Units	6 IALU, 2 IMULT, 4 FPALU, 2 FPMULT

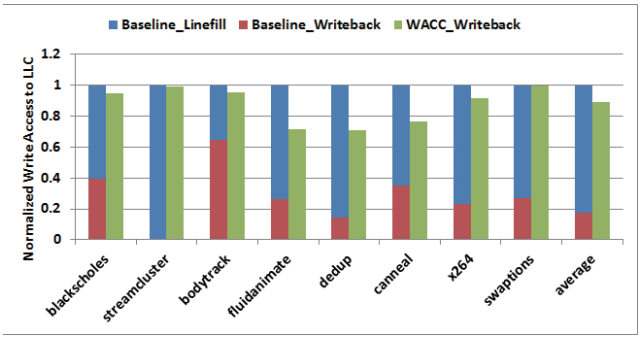

Fig. 4 The normalized number of the access to LLC of WACC protocol compared to the MOESI protocol.

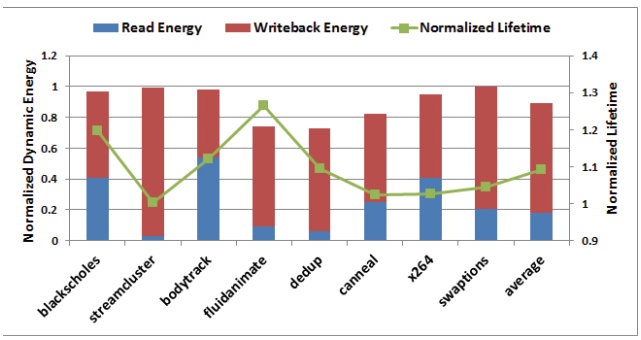

Fig. 5 The normalized dynamic energy consumption and lifetime of WACC compared to the baseline MOESI protocol.

to the LLC. The improvement of average write endurance in WACC protocol is 26.3% at maximum and 9.3% on average.

5. Conclusion

In this letter, we proposed a novel cache coherence protocol to eliminate useless write operations of LLC for multi-core system. Based on the analysis of the existing protocols, it was found that they generated useless write access to the LLC during the linefill operation. Thus, the LLC of our protocol which is called WACC only changes the cache states without storing the block data when it is arrived. This write policy reduced the number of write access to the LLC, which led to improvements in the energy consumption and lifetime. The simulation result showed that the reduction of maximum energy consumption in WACC protocol is 27.1% and lifetime extension is 26.3% at maximum in STT-RAM based LLC.

References

[1] M. Hosomi, H. Yamagishi, T. Yamamoto, K. Bessho, Y. Higo, K. Yamane, H. Yamada, M. Shoji, H. Hachino, C. Fukumoto, H. Nagao, and H. Kano, "A novel nonvolatile memory with spin torque transfer magnetization switching: Spin-ram," Electron Devices Meeting, 2005. IEDM Technical Digest. IEEE International, pp.459–462, 2005.

[2] S. Raoux, G.W. Burr, M.J. Breitwisch, C.T. Rettner, Y.C. Chen, R.M. Shelby, M. Salinga, D. Krebs, S.H. Chen, H.L. Lung, and C.H. Lam, "Phase-change random access memory: A scalable technology," IBM Journal of Research and Development, vol.52, no.4.5, pp.465–479, 2008.

[3] P. Zhou, B. Zhao, J. Yang, and Y. Zhang, "Energy reduction for stt-ram using early write termination," IEEE/ACM International Conference on Computer-Aided Design-Digest of Technical Papers, 2009. ICCAD 2009. pp.264–268, 2009.

[4] Y. Joo, D. Niu, X. Dong, G. Sun, N. Chang, and Y. Xie, "Energy- and endurance-aware design of phase change memory caches," Proc. Conference on Design, Automation and Test in Europe, pp.136–141, European Design and Automation Association, 2010.

[5] "The intel 64 and ia-32 architectures software developer's manual." http://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-software-developer-system-programming-manual-325384.pdf. accessed 3-Mar-2014.

[6] D.J. Sorin, M.D. Hill, and D.A. Wood, "A primer on memory consistency and cache coherence," Synthesis Lectures on Computer Architecture, vol.6, no.3, pp.1–212, 2011.

[7] C. Bienia, S. Kumar, J.P. Singh, and K. Li, "The parsec benchmark suite: Characterization and architectural implications," Proc. 17th International Conference on Parallel Architectures and Compilation Techniques, pp.72–81, 2008.

[8] N. Binkert, B. Beckmann, G. Black, S.K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D.R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M.D. Hill, and D.A. Wood, "The gem5 simulator," ACM SIGARCH Computer Architecture News, vol.39, no.2, pp.1–7, 2011.

[9] G. Sun, X. Dong, Y. Xie, J. Li, and Y. Chen, "A novel architecture of the 3D stacked MRAM l2 cache for CMPS," IEEE 15th International Symposium on High Performance Computer Architecture, 2009. HPCA 2009. pp.239–249, 2009.