ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014

Proceedings of International Conference On Global Innovations In Computing Technology (ICGICT'14)

Organized by

Department of CSE, JayShriram Group of Institutions, Tirupur, Tamilnadu, India on 6th & 7th March 2014

An Energy Improvement in Cache System by Using Write Through Policy

Vigneshwari.S[1]

PG Scholar, Department of ECE VLSI Design, SNS College of Technology, CBE-641035, India[1]

ABSTRACT: This project presents an energy efficient L2 cache memory using way tag information under write through policy. Many high-performance microprocessors employ cache write-through policy for performance improvement and at the same time achieving good tolerance to soft errors in on-chip caches. In this project, propose new cache architecture referred to as way tagged cache to improve the energy efficiency of write-through caches. By maintaining the way tags of L2 cache in the L1 cache during read operations, the proposed technique enables L2 cache to work in an equivalent direct mapping manner during write hits, which account for the majority of L2 cache accesses. This leads to significant energy reduction without performance degradation. The idea of way tagging can be applied to existing low-power cache design techniques to further improve energy efficiency

I. INTRODUCTION

Multilevel on-chip cache systems have been widely adopted in high-performance microprocessors. To keep data consistence throughout the memory hierarchy, two replacement policies are used widely, write through and write back policies are commonly employed. Under the write back policy, a modified cache block is copied back to its corresponding lower level cache only when the block is about to be replaced. While under the write through policy, all copies of a cache block are updated immediately after the cache block is modified at the current cache, even though the block might not be evicted. As a result, the write through policy maintains identical data copies at all levels of the cache hierarchy throughout most of their life time of execution.

1.1 Cache

The cache is a smaller, faster memory which stores copies of the data from the most frequently used main memory locations. As long as most memory accesses are cached memory locations, the average latency of memory accesses will be closer to the cache latency than to the latency of main memory. When the processor needs to read from or write to a location in main memory, it first checks whether a copy of that data is in the cache. If so, the processor immediately reads from or writes to the cache, which is much faster than reading from or writing to main memory. a translation look aside buffer (TLB) used to speed up virtual-to-physical address translation for both executable instructions and data.

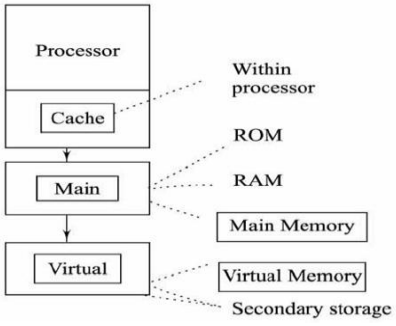

Fig:1.1 Flow of cache with in processor

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014

Proceedings of International Conference On Global Innovations In Computing Technology (ICGICT'14)

Organized by

Department of CSE, JayShriram Group of Institutions, Tirupur, Tamilnadu, India on 6th & 7th March 2014

Data is transferred between memory and cache in blocks of fixed size, called *cache lines*. When a cache line is copied from memory into the cache, a cache entry is created. The cache entry will include the copied data as well as the requested memory location now called a tag. When the processor needs to read or write a location in main memory, it first checks for a corresponding entry in the cache. The cache checks for the contents of the requested memory location in any cache lines that might contain that address. If the processor finds that the memory location is in the cache, a cache hit has occurred.

1.2 Organization Of Cache

There are three common ways to organize a cache . Fully associative cache, Direct mapped cache, set-associative cache. This project deal with the two mapping method direct map cache and set associative cache.

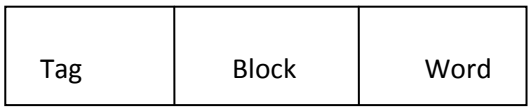

Fig: direct mapped cache

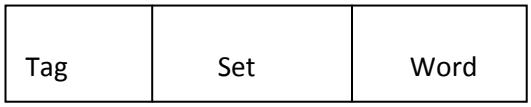

Fig: set associative mapped cache

In direct mapped cache, data access from main memory to cache is one to one mapping In a direct-mapped cache, any program block can be Placed in only one cache block. In a set-associative cache, cache blocks are divided into sets and a program block may be within any of the cache blocks in one set.

II. EXISTING SYSTEM

3.1 Two Level Cache

Many techniques have been developed to reduce cache power dissipation. Su et al partitioned cache data arrays into sub banks. During each access, only the sub bank containing the desired data is activated. Ghose et al further divided cache bit lines into small segmentations. When a memory cell is accessed only the associated bit line segmentations are evaluated. With the necessary software support, this cache can be configured as direct- mapping, two-way, or four-way set-associative according Another technique referred to as way concatenation was proposed by Zhang et al to reduce the cache energy in embedded systems. With the necessary software support, this cache can be configured as direct- mapping, two-way, or four-way set-associative according to the system requirements.

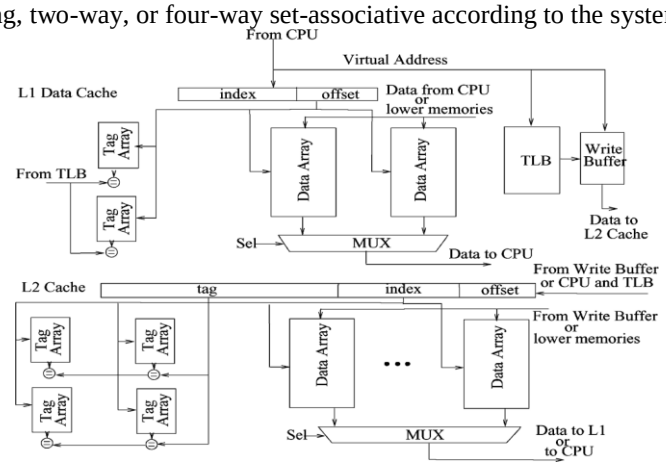

Fig: 3.1 two level cache architecture

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014

Proceedings of International Conference On Global Innovations In Computing Technology (ICGICT'14)

Organized by

Department of CSE, JayShriram Group of Institutions, Tirupur, Tamilnadu, India on 6th & 7th March 2014

3.2 Disadvantage of existing system

In the existing architecture Only the L1 data cache and L2 unified cache are shown as the L1 instruction cache only reads from the L2 cache. Under the write through policy, the L2 cache always maintains the most recent copy of the data. Power consumption and area is large.

III. PROPOSED SYSTEM

4.1 Way Tag Cache

In this project propose new cache architecture, referred to as way tagged cache, to improve the energy efficiency of write through cache systems with minimal area overhead and no performance degradation. A way tagged cache that exploits the way information in L2 cache to improve energy efficiency. It consider a conventional set-associative cache system when the L1 data cache loads/writes data from/into the L2 cache, all ways in the L2 cache are activated simultaneously for performance consideration at the cost of energy overhead.

Consider a two-level cache hierarchy, where the L1 data cache is write-through and the L2 cache is inclusive for high performance. It is observed that all the data residing in the L1 cache will have copies in the L2 cache

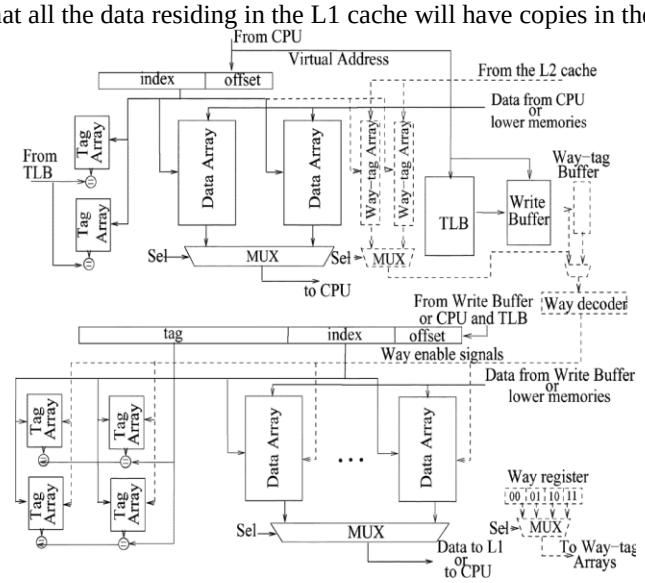

Fig: 4.1 Proposed way tagged cache

. In addition, the locations of these copies in the L2 cache will not change until they are evicted from the L2 cache. Thus, we can attach a tag to each way in the L2 cache and send this tag information to the L1 cache when the data is loaded to the L1 cache. By doing so, for all the data in L1 cache we will know exactly the locations (ways) of their copies in the L2 cache. During the subsequent accesses when there is a write hit in the L1 cache we can access the L2 cache in an equivalent direct-mapping manner because the way tag of the data copy in the L2 cache is available. As this operation accounts for the majority of L2 cache accesses in most applications, the energy consumption of L2 cache can be reduced significantly. The locations (i.e., way tags) of L1 data copies in the L2 cache will not change until the data are evicted from the L2 cache. The proposed way-tagged cache exploits this fact to reduce the number of ways accessed during L2 cache accesses. When the L1 data cache loads a data from the L2 cache, the way tag of the data in the L2 cache is also sent to the L1 cache and stored in a new set of way-tag arrays. These way tags provide the key information for the subsequent write accesses to the L2 cache.

4.2 Implementation Of Way Tagged Array

In the proposed way-tagged cache, each cache line in the L1 cache keeps its L2 way tag information in the corresponding entry of the way-tag arrays, where only one L1 data array and the associated way-tag array are shown for simplicity. When a data is loaded from the L2 cache to the L1 cache, the way tag of the data is written into the way-tag array. At a later time when updating this data in the L1 data cache, the corresponding copy in the L2 cache needs to be updated as well under the write-through policy.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014

Proceedings of International Conference On Global Innovations In Computing Technology (ICGICT'14)

Organized by

Department of CSE, JayShriram Group of Institutions, Tirupur, Tamilnadu, India on 6th & 7th March 2014

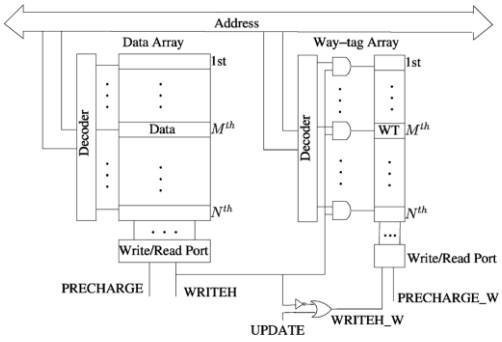

Fig: 4.2 Way tag array

The way tag stored in the way-tag array is read out and forwarded to the way-tag buffer together with the data from the L1 data cache. A control signal referred to as UPDATE is obtained from the cache controller. When the write access to the L1 data cache is caused by a L1 cache miss, UPDATE will be asserted and allow WRITEH_W to enable the write operation to the way-tag arrays (WRITEH=1, UPDATE=1).

WRITH	UPDATE	OPERATION
1	1	Write way tag array
1	0	Read way tag array
0	0	No access
0	1	No access

Table:1 Operations Of Way-Tag Arrays

4.3 Way Register

The way register provides way tags for the way-tag arrays. For a 4-way L2 cache, labels "00", "01", "10", and "11" are stored in the way register, each tagging one way in the L2 cache.

4.4 Way Tag Buffer

Way-tag buffer temporarily stores the way tags read from the way-tag arrays. The implementation of the way-tag buffer is shown in Fig. 4.4.

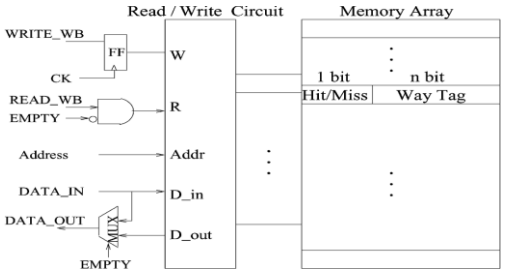

Fig: 4 .4 Way tag buffer

It has the same number of entries as the write buffer of the L2 cache and shares the control signals with it. Each entry of the way-tag buffer has bits, where is the line size of way-tag arrays. An additional status bit indicates whether the operation in the current entry is a write miss on the L1 data cache

V. CONCLUSION

This paper presents a new energy efficient cache technique for high performance microprocessors employing the write-through policy. The proposed technique attaches a tag to each way in the L2 cache. This way tag is sent to the way tag arrays in the L1 cache when the data is loaded from the L2 cache to the L1 cache. Using way tag of L2 data into L1 cache, the power and access time can be reduced.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014

Proceedings of International Conference On Global Innovations In Computing Technology (ICGICT'14)

Organized by

Department of CSE, JayShriram Group of Institutions, Tirupur, Tamilnadu, India on 6th & 7th March 2014

REFERENCES

1. A. Hasegawa, I. Kawasaki, K. Yamada, S. Yoshioka, S. Kawasaki, and P. Biswas, "Sh3: High code density, low power," *IEEE Micro*, vol. 15, no. 6, pp. 11–19, Dec. 1995.
2. C. Su and A. Despain, "Cache design tradeoffs for power and performance optimization: A case study," in *Proc. Int. Symp. Low Power Electron. Design*, 1997, pp. 63–68.
3. K. Ghose and M. B.Kamble, "Reducing power in superscalar processor caches using subbanking, multiple line buffers and bit-line segmentation,"in *Proc. Int. Symp. Low Power Electron. Design*, 1999, pp. 70–75.
4. K. Inoue, T. Ishihara, and K. Murakami, "Way-predicting set-associative cache for high performance and low energy consumption," in *Proc. Int. Symp. Low Power Electron. Design*, 1999, pp. 273–275.
5.G. Konstadinidis, K. Normoyle, S. Wong, S. Bhutani, H. Stuimer, T. Johnson, A. Smith, D.Cheung, F. Romano, S. Yu, S. Oh, V.Melamed, S. Narayanan, D. Bunsey, C. Khieu, K. J. Wu, R. Schmitt, A. Dumlao, M. Sutera, J. Chau, andK. J. Lin, "Implementation of a third-generation 1.1-GHz 64-bit microprocessor," *IEEE J. Solid-State Circuits*, vol. 37, no. 11, pp. 1461–1469, Nov. 2002.