International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2014): 5.611

Enhanced Way Tagged Cache Design Using Partial Tag Bloom Filter for Low Level (L2) Cache

Shaima Ibrahim[1], Sumi Babu[2]

[1]PG Scholar,ICET
[2]Assistant Professor, ICET

Abstract: ***High-performance microprocessors utilize cache write-through policy for performance improvement, at the same time achieving good tolerance to soft errors in on-chip caches. Write-through policy also consumes large power due to the increased access to caches in different level during write operation. The objective of this paper is to improve the energy efficiency of write-through caches as well as improving the access time with a new cache architecture referred as way tagged cache. Many high performance microprocessor designs have chosen the write-through policy by maintaining the way tags of L2 cache in the L1 cache during read operations, proposed method uses L2 cache to work in an equivalent direct-mapping manner during write hits, which account for the majority of L2 cache accesses. During read operation, the way tag of the L2 cache information is available in L1 cache. In the proposed method, partial tag partially matches the incoming address with the cache line address. If the partial tag gives matches then full tag comparison will be performed. If no match is found, then the cache line(s) is not associated with the incoming address. Due to this the way-tag technique enables L2 cache to work in an equivalent direct mapping manner during write hits when the cache data compared with the input data, and it leads to reduce the significant energy without performance degradation and the access time. While the corresponding way tag information is not available in the way-tag arrays for read misses, therefore all the ways in the L2 cache are activated simultaneously under read misses.***

Keywords: Multi level Cache, write-through policy, write- back policy

1. Introduction

On-chip cache has been used in microprocessor and it achieves high memory performance and low energy consumption. There are two operations associated with a cache memory-read and write operation. In memory hierarchy write-through and write back polices are commonly used. In a write back policy, if the data is updated to the cache, those data"s are not immediately updated to the main memory. Those data will be updated to the main memory only when the data is replaced from the cache. In write through policy, if the data is written (or) updated to the cache, immediately the same data will be updated to the main memory. Due to this during the life time execution the write-through policy keep the identical data copies.

High-performance microprocessors employ cache write-through policy for performance improvement at the same time achieving good tolerance to soft errors in on-chip caches. Due to the increased access to caches in different level during write operation write-through policy also consumes large power. Here an efficient cache design referred to as way-tagged cache with partial tag comparison which minimizes the access time without any performance degradation.

2. Cache Memory Basics

The memory hierarchy of a processor that includes the processor core, cache memory, RAM, hard disk and other lower levels of memory. Cache memory is called CPU memory, is a RAM (Random Access Memory) that can access the data more quickly as compared to the regular RAM. Sometimes it is typically integrated directly with the CPU chip or placed on a separate chip and connected with the CPU through a separate bus interconnect.

The cache is a smaller, faster memory which stores the replicated copies of the data from the frequently used main memory locations. The fast access to these instructions increases the overall speed of the program. Traditionally, it is categorized as "levels" that describe its closeness and accessibility to the processor: Level 1 (L1) cache, Level 2 (L2) cache, L3 and other lower levels; etc. L1 cache is relatively small but extremely fast, usually embedded in the CPU. L2 cache is larger as compared with L1; it may be located on the CPU or on a separate chip. L1 cache has data cache and instruction cache. L2 cache that has copies of all the data in the L1 cache. L3 cache is specialized memory to improve the performance of L1 and L2, has double the speed RAM. In the case of multicore processors, each core may have its own dedicated L1 and L2 cache, but share a common L3 cache. When an instruction is referenced in the L3 cache, it is typically elevated to a higher tier cache.

In two level cache system (i.e.: L1 and L2).If the write-back policy is adapted in L1 cache, it does not required to access the L2 cache. But if the write through policy is adapted in L1 cache, then for every write operation both two level cache (L1&L2 cache) needs to be accessed. Therefore write-through policy uses more write access in the L2 cache, and thereby increases more energy consumption of cache and which is the major issues in the caches.

3. Previous Work

To reduce the cache access time and energy consumption many techniques have been developed.

A. Traditional Cache Search Technique

Whenever the processor needs to fetch a data from the memory, it first looks into the cache memory. All the data in L1 cache that have a replica in L2 cache also. If the data is found in L1 then the location of the same data in L2 is found only through search all the data entries in L2. Thereby

access time is large enough in the case of finding the location of the same data.

B. Way Tagged Cache

Under the write-through policy, the way-tagged cache improves the efficiency of energy and increases the performance. In a two level cache hierarchy, whatever the data is residing in the L1 cache, the same data will be available in the L2 cache and the location of those data"s in the L2 cache will change only when they are deleted (or) removed from the L2 cache. When the L1 data cache reads a data from the L2 cache and in each way of L2, the way-tag is attached. So the way-tag of the data will be sent to the L1 cache and it will be updated (or) copied in a way-tag arrays. By doing these, the way-tag of information"s is provided for the successive write accesses to the L2 cache. For every read and write operations in the L1 cache, L2 cache need to be accessed. In which, the way-tag cache performs different operations for read and write access. The correct locations (or) ways are available in L2 cache, for all the data in the L1 cache. During the write hit in the L1 cache, by using the way-tag technique the way-tag of the data copy can be accessed in the L2 cache and it access the L2 cache. . If there is write hit in the L1 cache, the L2 cache selects only one way and that will be activated. Because the way-tag is attached in each way of L2, due to this it provides information to the way-tag array. From the way-tag array, we can get the desired way of data. If there is write-miss in the L1 cache means, the requested data is not available with the L1 cache (i.e.) if the data is not there in the L1 cache the desired information will not be available in way-tag arrays.

So, under the write miss in all the ways the L2 cache can be accessed simultaneously. Due to this it consumes more energy consumption. During the read hit, the requested data is available in the L1 cache, therefore L2 cache need not to be accessed. If there is read miss, the requested data is not available in the L1 cache. Therefore, the L2 cache needs to be accessed simultaneously in all the ways.

4. Proposed Design

In the previous work-way tagged cache, the access time and efficiency have become much better which when compared with the traditional cache search technique.

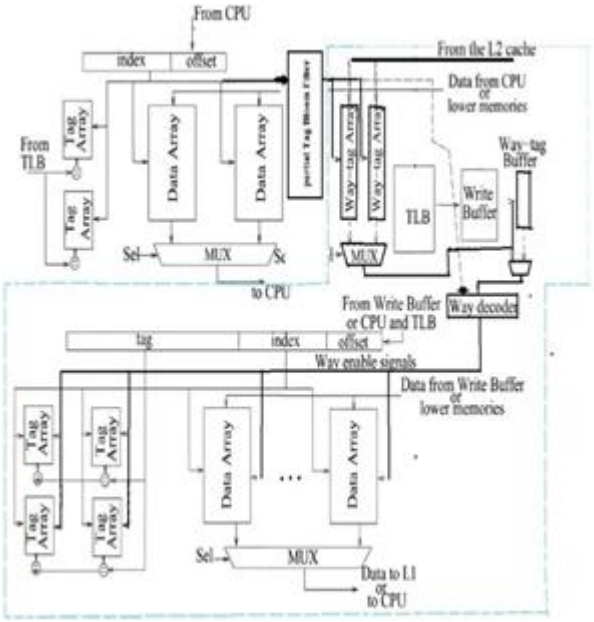

Figure 1: Proposed Way Tagged Cache Architecture

In the proposed design, way tagged cache architecture with partial tag matching, uses Count Bloom Filter (CBF). By using the proposed way-tag cache, under the write operation it access only fewer ways and it reduces the memory energy consumption.

The way tag architecture includes many new components, shown in Fig.1. - way tag array, way decoder, way tag buffer, and way register. To each way in the cache line the tag is attached and this tag information of L2 cache are updated in the way tag array, located in the L1 data cache. Write buffer uses both write through and write back policies to enhance the performance. Registers are used as a storage device and way registers are used for storing the way tags.

C. Way Tag Array

In the way-tagged cache, each cache line in the L1 cache keeps its L2 way tag information in the corresponding entry of the way-tag array. The way tag of the data is written into the way-tag array whenever a data is loaded from the L2 cache to the L1 cache .At a later time whenever the data is updated with the L1 cache, the corresponding replica in the L2 cache wants to be updated under the write-through policy. The way tag stored in the way-tag array is read out and together with the data from the L1 data cache the data is forwarded to the way- tag buffer. In the proposed design, each cache line in the L1 cache keeps its L2 way tag information in the corresponding entry of the way-tag arrays, as shown in Fig. 4, 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

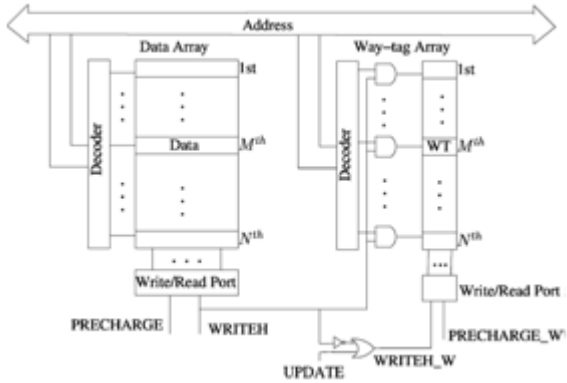

Figure 2: Way Tag Array

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. 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. Note that the data arrays in the L1 data cache and the way-tag arrays share the same address as the mapping between the two is exclusive. The write/read signal of way-tag arrays, WRITEH_W, is generated from the write/read signal of the data arrays in the L1 data cache as shown in Fig. 2.

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, see TABLE I). If a STORE instruction accesses the L1 data cache, UPDATE keeps invalid and WRITE_W indicates a read operation to the way-tag arrays (WRITEH=1, UPDATE=0). During the read operations of the L1 cache, the way-tag arrays do not need to be accessed and thus are deactivated to reduce energy overhead. To achieve this, the word line selection signals generated by the decoder are disabled by WRITEH (WRITEH=0, UPDATE=1/0) through AND gates. The above operations are summarized in Table I.

To avoid performance degradation, the way-tag arrays are operated in parallel with the L1 data cache. Due to their small size, the access delay is much smaller than that of the L1 cache. On the other hand, the way-tag arrays share the address lines with the L1 data cache. Therefore, the fan-out of address lines will increase slightly. This effect can be well-managed via careful floor plan and layout during the physical design. Thus, the way-tag arrays will not create new critical paths in the L1 cache.

Table 1: Operation of Way Tag Array

Write	Update	Operation
1	1	Write into Way Tag Array
1	0	Read from Way tag Array
0	x	No Access

D. Way Tag Buffer

Way tag buffers temporarily stores the way tag information from the way-tag array. Way-tag array and way-tag buffer shares the same control signal and both have the same number of entries. Way-tag buffer has n+1 bi t for each entry and in which n is the line size of way-tag arrays. Status bit is used to determine the operation in the current entry is a write-miss on the L1 data cache. The way-tag information is not available in the way-tag array in the case of write miss and therefore all the ways of the way-tags will be activated.

Similar to the write buffer of the L2 cache, the way-tag buffer has separate write and read logic in order to support parallel write and read operations. The write operations in the way-tag buffer always occur one clock cycle later than the corresponding write operations in the write buffer. This is because the write buffer, L1 cache, and way-tag arrays are all updated at the same clock cycle when a STORE instruction accesses the L1 data cache (see Fig. 2). Since the way tag to be sent to the way-tag buffer comes from the way-tag arrays, this tag will be written into the way-tag buffer one clock cycle later. Thus, the write signal of the way-tag buffer can be generated by delaying the write signal of the write buffer by one clock cycle, as shown in Fig. 3.

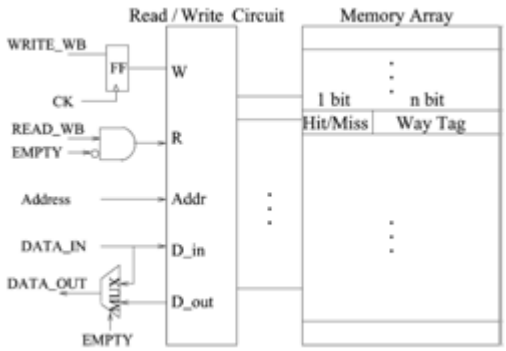

Figure 3: Way Tag Buffer

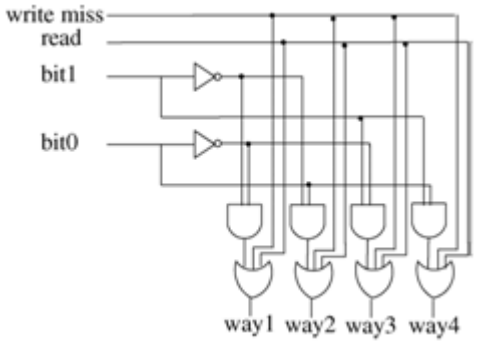

Figure 4: Way Decoder

E. Way Decoder

The way decoder is used to decode the way tags and selects only the correct way in the L2 cache. The way tag array of the line size is n=log 2 N bits. If there is write hit in the L2 cache the way decoder decodes the way tags and activates only in one way. In L2 cache tag and data arrays are decoded by the way decoder and it can be decoded at the same time. If there is write miss (or) read miss in the L1 cache, all the ways of the L2 cache is need to be selected. The operation mode of the way decoder can be determined by the read miss/write miss of the two signals. If the read access is sent to the L2 cache, then read signal will be „1".

The way decoder operates simultaneously with the decoders of the tag and data arrays in the L2 cache. For a write miss or a read miss in the L1 cache, we need to assert all way-enable signals so that all ways in the L2 cache are activated. To achieve this, the way decoder can be implemented by the circuit shown in Fig. 4. Two signals, *read* and *write miss*, determine the operation mode of the way decoder. Signal *read* will be "1" when a read access is sent to the L2 cache. Signal *write miss* will be "1" if the write operation accessing the L2 cache is caused by a write miss in the L1 cache.

F. Way Register

The way tag register stores the way tags and their information for the way tag array. In L2 cache four way tag arrays are available. The tag arrays can be indicated as "00","01","10","11",these are available in way register. When the data is loaded from the L2 cache to L1 cache, according to the way-tag in the register stored in the way-tag array.

G. Partial Tag Bloom Filter

Partial tag compares the part of the incoming address with the Cache lines in the corresponding address**.** If there is no match then the corresponding cache address is not

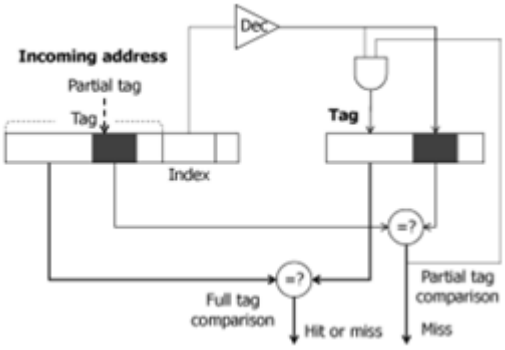

Figure 5: Partial Tag Comparison

associated with incoming address. If the comparison of partial tag gives match, then full tag comparison will be performed.

Before the tag structure enter into each cache way is accessed, first the Bloom filter per cache way is looked up. If the Bloom filter indicates zero, then tag comparison for the cache way is avoided. Due to this we can reduce the most of the energy consumption during the tag comparison. Due to this during the tag comparison operation, it reduces the energy consumption.
If the partial tag matches then only the full tag comparison will be performed. If the full tag comparison then the read hit is obtained. If the partial tag does not match then no need to check the entire bits of the cache data. So the access time can be reduced along with the energy consumption.

5. Results

Several cache search techniques were introduced. The proposed design is utilized in the case of a large number of cache entries, it is easier to search and with less access time and energy implemented in VHDL, simulated with ModelSim and synthesized with Xilinx. In order to evaluate the improvement of the proposed design, the energy consumption and latency are compared with the traditional designs. The proposed design uses only one third of energy of the conventional way tagged cache architecture. The proposed design does not need to check the entire address of all cache data in the case of cache miss, which increases more energy and time.

Table 2: Comparison Results

	Way Tagged Cache	Cache with Partial Tag Comparison
Energy	n*Energy for 1 bit search	Energy for 1 bit search
Maximum combinational path delay	26.487ns	24.36ns

6. Simulation Results

The proposed Way Tagged Cache with Partial Tag Comparison is simulated with ModelSim and synthesized with Xilinx. The simulation results for different inputs.

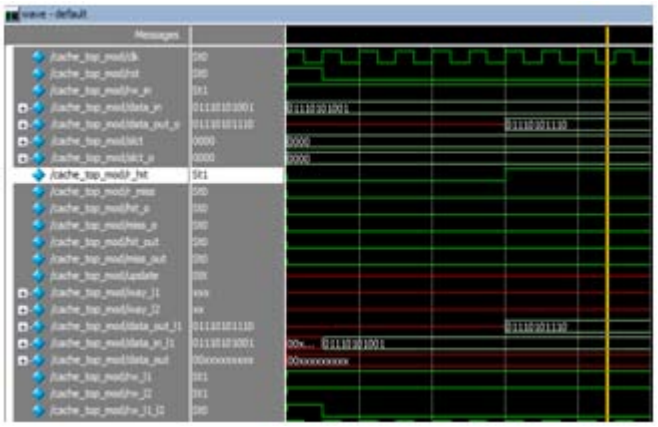

Figure 6: Read operation and data HIT in L1

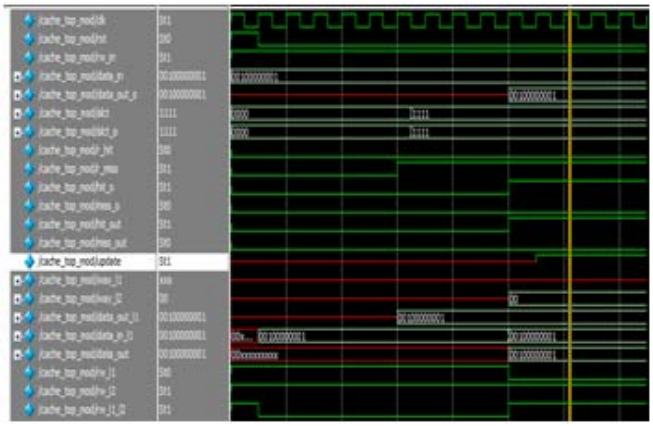

Figure 7: Read operation and data MISS in L1 and found in L2

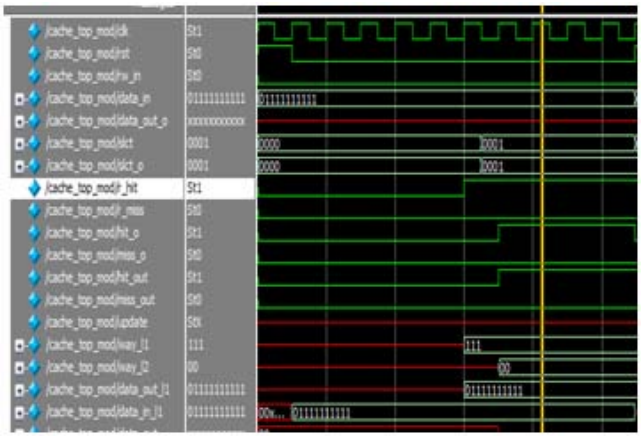

Figure 8: Write Operation into L1 and L2 with tag information

7. Conclusion

The improved architecture design and implementation for the way tagged cache architecture has been proposed. Several critical design issues for the proposed way tagged cache architecture- with partial tag comparison includes area and complexity but these issues are resolved with performance enhancement and time access, since the proposed design requires less time access without any performance degradation.

References

[1] J. Dai and L. Wang, "Way-tagged cache: An energy efficient L2 cache architecture under write through policy," in *Proc. Int. Symp. Low Power Electron. Design*, 2009, pp. 159–164.

[2] X. Vera, J. Abella, A. Gonzalez, and R. Ronen, "Reducing soft error vulnerability of data caches," presented at the Workshop System Effects Logic Soft Errors, Austin, TX, 2007.

[3] Vineeta Vasudevan Nair, "Way-Tagged L2 Cache Architecture in Conjunction with Energy Efficient Datum Storage", in *International Journal of Electronics Communication and Computer Technology (IJECCT)* 2014, volume 4, pp. 543-548.

[4] C. Zhang, F. Vahid, and W. Najjar, "A highly-configurable cache architecture for embedded systems," in *Proc. Int. Symp. Comput. Arch.*, 2003, pp. 136–146.