iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 3 **Issue:** V **Month of publication:** May 2015

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)

A Way Early Tag Access with Partial Tag Comparison Technique for Reducing Power Consumption of L1 Data Caches

Neetha Anjelin D'sa[1], Manjunath Badiger[2]
[1]Student, Dept.of ECE, SDIT Mangalore (India)
[2]Assistant Professor, Dept.Of ECE, SDIT Mangalore (India)

Abstract —In this paper, we propose a new cache design technique, referred to as early tag access (ETA) cache, to improve the energy efficiency of data caches in embedded processors. The proposed technique performs ETAs to determine the destination ways of memory instructions before the actual cache accesses. It, thus, enables only the destination way to be accessed if a hit occurs during the ETA. The proposed ETA cache can be configured under two operation modes to exploit the trade-offs between energy efficiency and performance. It is shown that our technique is very effective in reducing the number of ways accessed during cache accesses. This enables significant energy reduction with negligible performance overheads. Simulation results demonstrate that the proposed ETA cache achieves over 52.8% energy reduction on average in the L1 data cache and translation look aside buffer. Compared with the existing cache design techniques, the ETA cache is more effective in energy reduction while maintaining better performance.
Keywords- Cache, Low power.

I. INTRODUCTION

Reducing power consumption in cache memory is a critical problem for embedded processors that target lowpower applications. It was reported that on-chip caches could consume as much as 40% of the total chip power, Furthermore, large power dissipation could cause other issues, such as thermal effects and reliability degradation. This problem is compounded by the fact that data caches are usually performance critical. Therefore, it is of great importance to reduce cache energy consumption while minimizing the impact on processor performance. Many cache design techniques have been proposed at different levels of the design abstract to exploit the tradeoffs between energy and performance. As caches are typically set-associative, most microarchitectural techniques aim at reducing the number of tag and data arrays activated during an access, so that cache power dissipation can be reduced. Phased caches access tag arrays and data array in two different phases. Energy consumption can be reduced greatly because at most only one data array corresponding to the matched tag, if any, is accessed. Due to the increase in access cycles, phased caches are usually applied in the lower level memory, such as L2 caches, whose performance is relatively less critical. For L2 caches under the write through policy, a way-tagging technique sends the L2 tag information to the L1 cache when the data is loaded from the L2 cache. During the subsequent accesses, the L2 cache can be operated in an equivalent direct-mapping manner, thereby improving energy efficiency without incurring performance degradation.

II. CONVENTIONAL ARCHITECTURE

Many cache design techniques have been proposed at different levels of the design abstract to exploit the tradeoffs between energy and performance. As caches are typically set-associative, most micro architectural techniques aim at reducing the number of tag and data arrays activated during an access, so that cache power dissipation can be reduced. Phased caches access tag arrays and data arrays in two different phases. Energy consumption can be reduced greatly because at most only one data array corresponding to the matched tag, if any, is accessed. Due to the increase in access cycles, phased caches are usually applied in the lower level memory, such as L2 caches, whose performance is relatively less critical. For L2 caches under the write through policy, a way-tagging technique sends the L2 tag information to the L1 cache when the data is loaded from the L2 cache. During the subsequent accesses, the L2 cache can be operated in an equivalent direct-mapping manner, thereby improving energy efficiency without incurring performance degradation. To reduce the energy consumption of L1 caches, way predicting techniques make a prediction on the tag and data arrays that the desired data might be located. If the prediction is correct, only one way is accessed to complete the operation;

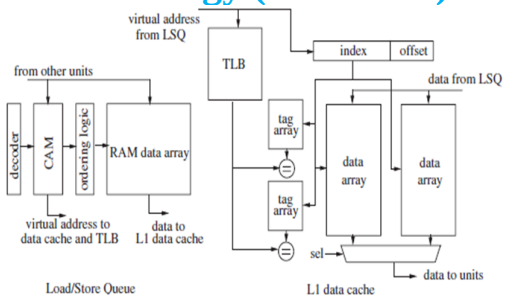


Figure 1: Conventional architecture of LSQ and L1 data cache**.**

To reduce conflict misses, set-associative architectures are commonly employed in cache design. In a two-way set-associative cache and TLB, where the tag and data arrays are the two major components. In the conventional L1 data cache, all tag arrays and data arrays are activated simultaneously for every read/write access to reduce the access latency. Typically, the latency of the L1 data cache is one clock cycle, though this latency could be higher in deeply pipelined processors. On the other hand, accesses to the tag arrays can always be finished in one cycle. Due to the temporal/spatial locality inherent in various programs, data will stay for a while once they have been brought into the data cache. This implies that the tag arrays will keep their contents unchanged except when a cache miss occurs. Figure shows a portion of the pipeline between the address generation stage and the memory stage in a typical embedded processor, where the LSQ and L1 data cache are accessed in series to take advantage of load forwarding for reducing cache traffic and energy consumption. A load/store instruction will be sent to the LSQ before being issued to the data cache. Meanwhile, the instruction is compared with the existing ones in the LSQ to determine whether the instruction can be issued to the data cache at the next clock cycle. If not, the instruction will stay in the LSQ stage for more than one clock cycle. From this architecture, observe that the memory address of a load/store will be available in the LSQ stage for at least one clock cycle before being issued to the data cache, while the access to the tag arrays can be finished in one cycle and 2) due to the temporal/spatial locality, the tag arrays will not be updated until a cache miss occurs. Since in the conventional architecture, the destination way of a memory instruction cannot be determined before the cache access stage, all the ways in the L1 data cache need to be activated during a cache access for performance consideration at the cost of energy consumption.

III. BASE PAPER PROPOSED

In a conventional set-associative cache, all ways in the tag and data arrays are accessed simultaneously. The requested data, however, only resides in one way under a cache hit. The extra way accesses incur unnecessary energy consumption. In this section new cache architecture referred to as ETA cache will be developed. The ETA cache reduces the number of unnecessary way accesses, thereby reducing cache energy consumption. To accommodate different energy and performance requirements in embedded processors, the ETA cache can be operated under two different modes: the basic mode and the advanced mode. accesses the L1 data cache at the cache access stage (which may happen several clock cycles later if some instructions in the LSQ have a higher priority than Inst1), data1 actually resides in the way1 of the L1 data cache due to a cache miss that happens between the LSQ stage and cache access stage of the instruction Inst1. In this case, a cache coherence problem will occur if Inst1 simply uses its early destination way to access the L1 data cache.To avoid this problem, the ETA cache in the basic mode requires the memory instruction to access all the ways in the actual tag arrays during the cache access stage. During the same time, the data arrays are also accessed in parallel using the early destination way only (as the actual destination way is not available yet) to reduce energy consumption while maintaining performance. The destination way obtained from the actual tag arrays is then compared with the early destination way to detect any cache coherence problem. If a cache coherence problem is detected, the accessed data from the data arrays is discarded and an additional access is performed. This occurs rarely. Therefore, only minor performance degradation is incurred, which is typically acceptable for embedded processors. Note that another method to resolve the above cache coherence problem is to only access the predicted early destination way during the cache access stage. If the prediction is incorrect, simultaneous tag-data lookup will be performed to obtain the desired data from the data arrays. This solution saves energy in accessing tag arrays but consumes more energy during the data array access when the prediction is incorrect. Since data arrays consume much larger energy than tag arrays choose to use the first method in this paper. However, if the re-access rate is extremely small, the second method may enable one percentage additional energy savings due to the reduction in accessing the tag arrays. Thus, most memory instructions can utilize their early destination ways

determined at the LSQ stage to access the L1 data cache.

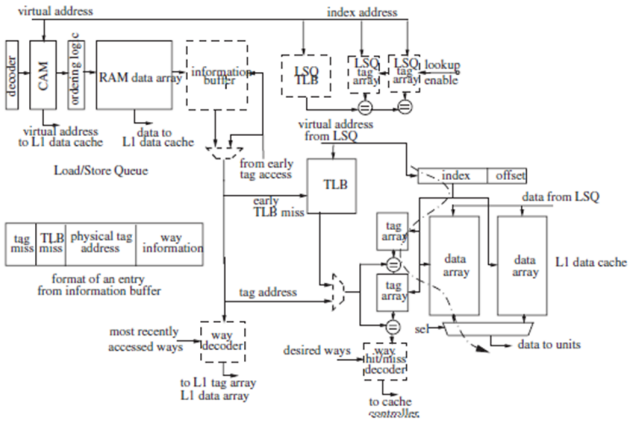


Figure 2: Proposed ETA cache (blocks in dash line are new components and dotted line is the timing critical path).

IV. MODIFICATION PROCESS

Partial Tag Comparison is comparing a small part of two different tags, instead of comparing the entire tag address for cache hit. The Bloom filter is utilized to check the approximate non membership of a set. When applied to reducing tag comparisons, each cache way is equipped with a Bloom filter. A query to the Bloom filter (e.g., "is address 0×100 in the cache way?") gives either of two results: negative (definite nonexistence) and positive (likely existence). Note that a negative result from the Bloom filter guarantees nonexistence, i.e.,a cache way miss. Thus, before the tag structure in each cache way is accessed, first the Bloom filter per cache way is looked up. If the Bloom filter indicates nonexistence, then tag comparison for the cache way is avoided, thereby saving the energy that would have been consumed in tag comparison.

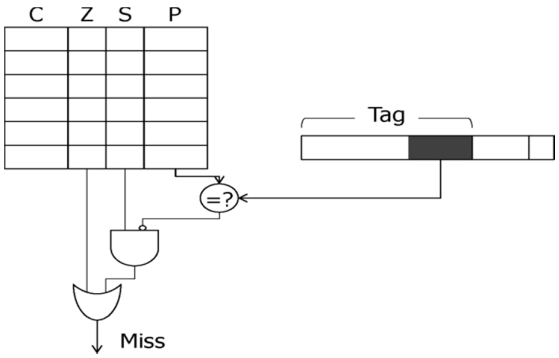


Figure 3: Modification using Bloom Filter.

We use a tag comparison technique with bloom filter. In Bloom filter the tag address is grouped according to the unchanged MSB Bits which have different combinations depends on the LSB bits and given an address in the tag array. When a Tag address is given the filter divides the MSB and LSB bits, first the MSB is compared and activate a particular address in the bloom filter TAG array then the comparison is done with the LSB bit in the array once the Cache hit occurs particular data from the L1 cache is given to the CPU and L1 cache. If the cache Miss occurs the Tag address is then given to the higher level of cache. A Bloom Filter entry has the tuple , where C is the counter, Z is the zero flag, S is the singleton flag, and P is the partial tag. The size of the partial tag is small, 3 bits. Thus, compared to the original Bloom filter, the partial tag enhanced Bloom filter has an overhead of 4 bits (including the S flag) per entry. The Bloom filter is utilized to check the approximate non-membership of a set. When applied to reducing tag comparisons, each cache way is equipped with a Bloom filter. For our modification, we show that the partial comparison method can filter out most of the unmatched tag comparisons. In the first step, the partial tag is compared with that of the incoming address. If the comparison does not yield a match, the remainder of the tag does not need to be compared; this saves energy because the corresponding cache way does not include the incoming address. If the partial tag comparison yields a match, then the remainder of the tag is compared. In the proposed architecture the virtual address is given

to the TLB, it generates the corresponding Tag address. The TAG address is compared with the TAG array in L1 cache using Pre computational logic. If the Cache hit occurs corresponding Data array is selected and the data is given to the CPU. If the cache Miss occurs then the tag address is given to the L1 cache.

V. COMPARISON WITH THE RELATED WORK

We compared the energy reduction and performance of our technique with related work, such as the way-predicting cache and the way-halting cache . All the cache designs have the same configuration, such as size and associativity. The energy results were obtained using the similar energy model from CACTI 5.3 for the same process technology. We have also compared the performance in terms of the CPI for all the cache designs using Simple scalar. Fig below compares five

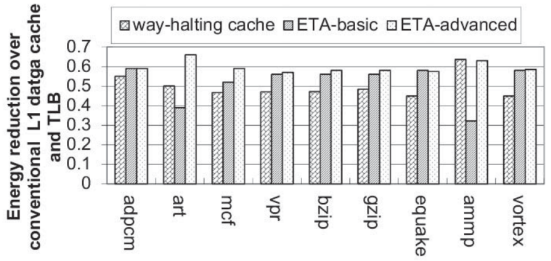


Figure 4: Comparison of energy reduction between the ETA cache and the way-halting cache.

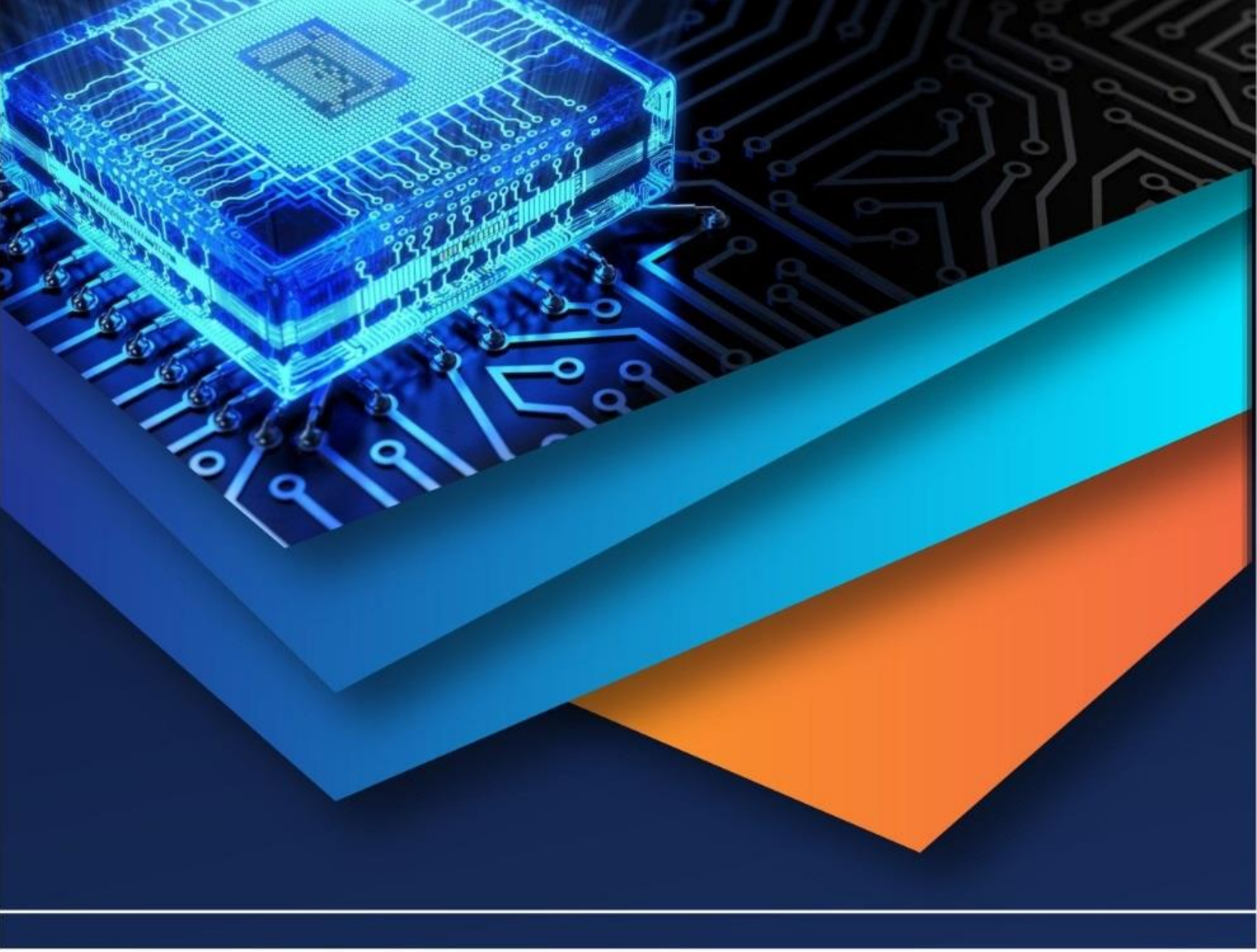
Figure shows the comparison with the way-halting cache with the same cache configuration. Under the basic mode, the proposed ETA cache outperforms the way-halting cache in all the benchmarks except for art and ammp. The reason is that the cache miss rates of these two benchmarks are high. Under the basic mode, the proposed technique accesses the data cache in the same way as the conventional cache when there is a cache miss. Thus, large energy overhead is resulted if the cache miss rate is elevated. On the other hand, the ETA cache under the advanced mode achieves better or equal energy reduction across all the benchmarks as compared with the way halting cache. This is because the ETA cache does not need to access the data arrays under a cache miss in the advanced mode. We have also found that the ETA cache achieves similar performance as the way-halting cache. In addition, similar energy-performance tradeoffs were observed under different cache configurations for the three cache designs.

VI. CONCLUSION

This paper presented a new energy-efficient cache design technique for low-power embedded processors. The proposed technique predicts the destination way of a memory instruction at the early LSQ stage. Thus, only one way needed to be accessed during the cache access stage if the prediction is correct, thereby reducing the energy consumption significantly. By applying the idea of phased access to the memory instructions whose early destination ways cannot be determined at the LSQ stage, the energy consumption can be further reduced with negligible performance degradation Simulation results demonstrated the effectiveness of the proposed technique as well as the performance impact and design overhead. While our technique was demonstrated by a L1 data cache design future work is being directed toward extending this technique to other levels of the cache hierarchy and to deal with multithreaded workloads.

REFERENCES

[1] *Intel XScale Microarchitecture*, Intel, Santa Clara, CA, USA, 2001.
[2] C. Zhang, F. Vahid, and W. Najjar "A highly-configurable cache architecture for embedded systems," in *Proc. 30th Annu. Int. Symp. Comput.Archit.*, Jun. 2003, pp. 136–146.
[3] S. Segars, "Low power design techniques for microprocessors," in *Proc. Int. Solid-State Circuits Conf. Tuts.*, Feb. 2001.
[4] S. Manne, A. Klauser, and D. Grunwald, "Pipline gating: Spculation conrol for energy reduction," in *Proc. Int. Symp. Comput. Archit.*, Jun.–Jul. 1998, pp. 132–141.
[5] M. Gowan, L. Biro, and D. Jackson, "Power considerations in the design of the alpha 21264 microprocessor," in *Proc. Design Autom. Conf*, Jun. 1998, pp. 726–731.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY