A Low Power Highly Associative Cache for Embedded Systems

Chuanjun Zhang
Department of Computer Science and Electrical Engineering
University of Missouri-Kansas City
zhangchu@umkc.edu

***Abstract*—Reducing energy consumption is an important issue for battery powered embedded computing systems. Content Addressable Memory (CAM)-based Highly-Associative Caches (HAC) are widely used in low power embedded microprocessors. The CAM tag is costly in power, access time, and area. We have designed a Low Power Highly Associative Cache (LPHAC) whose tag is partially implemented by using CAM, while the remaining tag is implemented by using SRAM. The experimental results from 10 MediaBench and all 26 SPEC2K benchmarks show the proposed LPHAC exhibits almost the identical miss rate as a traditional HAC. At the same time, it consumes 27% less per cache access power and 1.6% less area with faster access time.**

1. Introduction

Energy consumption is a major concern in many embedded computing systems. The microprocessor and memory consume a significant portion of the total energy of an embedded system. Several studies have shown that cache memories account for about 40% [3][12] of the total energy consumed in a microprocessor, thus an energy efficient cache architecture is a critical issue in the design of microprocessors for embedded systems.

Microprocessors designed for embedded systems are typically not equipped with a level two cache, which is widely available for high performance processors. Accessing off-chip memory is both time consuming and energy costly due to the high capacitance of the off-chip buses and the large storage of the off-chip memory. Therefore, reducing the miss rate of a level-one cache for embedded system microprocessors can greatly reduce the total power consumption.

The CAM-based HAC [3][9] is specifically designed for low power embedded systems where performance (cache access time) may be traded for low energy consumption. The CAM HAC reduces energy consumption of an embedded system through two main organizational techniques. One is to aggressively partition the cache memory into small subbanks, typically 1kB per bank. The small size of the subbank reduces the energy per cache access. The other is the high associativity. Typically, a 32-way cache is implemented in one subbank. High associativity greatly reduces the miss rate and the accesses to the off-chip buses and memory, which are both power and performance costly.

The problem of a traditional HAC design is the CAM tag consumes a significant portion of the total per-cache access power. CAM consumes 5-10 times more power than a same-sized SRAM [2]. Reducing the power consumption of the CAM-based tag is an important issue in low power HAC design.

The contribution of our technique is that we propose a low power CAM HAC (LPHAC) design that uses eight bits instead of 24 bits of CAM-based tag. The remaining 16 bits of the tag are implemented using SRAM, which is more efficient in area and power than CAM. We also show the tradeoff of the number of CAM tag bits used with the performance overhead in terms of hit rate. Using execution driven simulations from 10 Mediabench [4] and all 26 SPEC2K [10] benchmarks, we demonstrate that the miss rate of the proposed LPHAC remains almost the same with the original HAC while consuming 27% less per cache access power and 1.6% less area with faster access time.

This paper is organized as follows. Section 2 is a brief review of the related work. Section 3 describes traditional HAC architecture. Section 4 is the design of the proposed low power HAC. Section 5 presents experimental results. We analyze the LPHAC in Section 6 and conclude in Section 7.

2. Related Work

Substantial research has been conducted to reduce the energy consumption of HAC for embedded computing systems using organization and circuit's techniques.

Organization techniques include way prediction [11] and way memorization [5], which reduces access to the power costly CAM-based tags. Way-prediction HAC can skip the accesses of the CAM-based tag when the prediction of the next accessed cache way is correct. A simple last-used prediction technique can achieve an 86% correct prediction.

Way memorization cache records the last accessed way in a counter. When the identical way is next accessed, the tag can be skipped to save energy. Since the well-known locality exists in both data and instructions, way memorization can save 21% of the power. Compared to the proposed LPHAC, both way prediction and way memorization require extra hardware, while LPHAC reduces hardware from the traditional HAC design. Both way prediction and way memorization can be combined with the proposed LPHAC design to further reduce power consumption.

Circuit techniques can also be used to reduce the power consumption of CAM. Pipelined CAM [7] breaks the match lines into pipelined stages. Since mismatches typically happen in the early stages, the pipelined CAM reduces power through halting additional searching operations in other pipeline stages. The power savings for a 1024×144-bit pipelined CAM is 60% compared to a traditional CAM, which is still higher than a same-sized SRAM. The CAM size for HAC caches, however,

1-4244-9707-X/06/$20.00 ©2006 IEEE

is typically 32×24-bit. To fill the CAM pipeline, the CAM should be accessed in each cycle. However, for typical embedded processors, which use single-issue in-order core, one cache block (which may contain eight instructions) is enough for eight processor cycles. In this case, the processor may access the instruction cache every eight cycles when instruction locality is good, which diminishes the benefits of using pipelined CAM. Our technique uses a different method that is orthogonal with the pipelined CAM.

Techniques that use both circuit and organization techniques include serially accessed [2] and way-halting [13] cache. Serially accessed CAM may prolong the cache access time. Way-halting cache needs a specially designed CAM, which may not be easily available for embedded system designs. The proposed LPHAC reduces power consumption by using less CAM without any hardware overhead, special circuit design techniques, or special libraries.

3. Traditional HAC Design

3.1 Organization

Figure 1 shows the organization of a traditional 32-way HAC at a size of 8kB with a line size of 32 bytes. Two organizational techniques are employed to reduce power consumption. One is cache memory partition, the other is high associativity.

The cache memory is aggressively partitioned into eight subbanks with a size of 1kB for each subbank. Only one subbank is activated during one cache access to reduce the per cache access energy. This low power organization comes in exchange for performance (cache access time) overhead, since the wordline activation and bit-line pre-charging of both CAM-based tag and SRAM data of subbanks cannot start (to save power) until the subbank decoding finishes. This performance overhead is intolerable in high performance level-one cache designs where the subbank decoding of a high performance cache is conducted in parallel with the index decoding. The index decoding takes longer time than the subbank decoding and so hides the latency of subbank decoding. It should be noted that cache memory for high performance level-one caches is also partitioned into several subbanks to balance the power dissipation, access time, and area. The number of subbanks, however, is much lower than that of HAC, e.g., a same-sized high performance level cache has only four subbanks [8] instead of eight for HAC.

The HAC uses a fully associative cache, which is efficient in reducing misses. However, continuing to increase associativity higher than four or eight is not important, since for most applications, the miss rate reduction for associativity higher than eight or even four is diminishing. Implementing a 4-way cache on a 1kB cache block, however, is cumbersome and may not be as efficient as a fully associative cache using CAM to do the parallel searching. A size of 32 rows of CAM can be implemented efficiently in terms of access time and power [6]. Therefore, a fully associative cache is adopted for the low energy embedded system design.

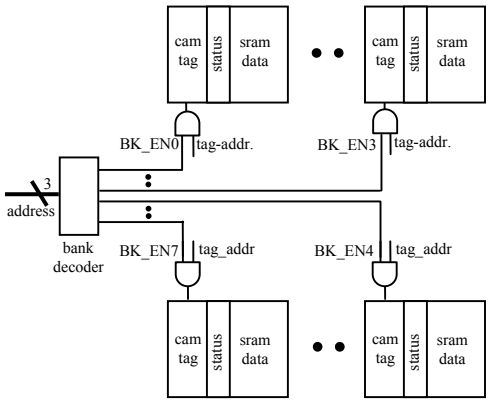

Figure 1: Traditional highly associative cache with eight subbanks. Each subbank is a fully associative cache. BK_EN0 ~ BK_EN7 are used to control the activation of the subbanks.

3.2 The Problem

Figure 2 shows the organization of the one subbank of the traditional HAC (a) and the proposed LPHAC (b). The problem with the CAM-based HAC is the tag consumes a significant portion of the total energy per cache access. Since all bit-lines of a CAM subbank are precharged. During a cache access, at most, one cache line matches the desired address; therefore, all other bit-lines and match lines have to be discharged, which makes CAM energy costly.

4. The LPHAC Design

4.1 Observation

We have observed that the function of the CAM tag in a traditional HAC is threefold. First, the tag serves as a full tag comparison and verifies a cache hit. Second, the CAM tag serves as a decoder whose output drives one cache line when there is a cache hit. Lastly, the CAM tag provides the cache an opportunity to choose a victim from the 32 cache lines. This happens during a cache miss when none of the 32 tags stored in the CAM tag matches the desired address.

For the first function, the entire tag is required to make a full comparison. For the second function, however, decoding 32 cache lines does not need all the 24 tag bits. In fact, 5-bit CAM is enough to distinguish the 32 lines. For the third function, a victim can be selected from the 32 cache lines for a cache miss, since none of the 32 tags matches the desired address. It is well known that cache misses occur mostly on the low order bits of

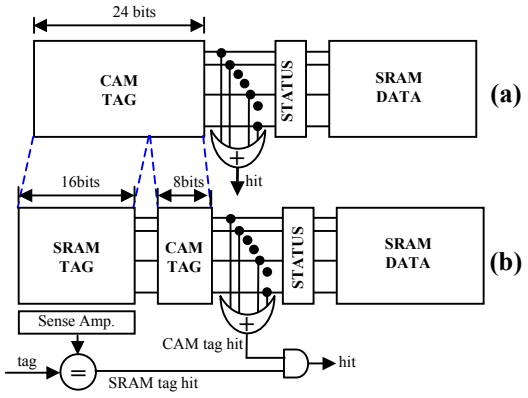

Figure 2: Organization of one subbank of original HAC (upper) and the proposed LPHAC (lower).

the tags [13]. Therefore, we may not need all the tag bits to fulfill the third function. We may achieve the similar miss rate reductions by using just part of the tag to find the optimal victim for a cache miss.

4.2 Organization

Based on the above observation, we propose a new organization as shown in Figure 2 (b). We divide the original 32×24 CAM-based tag into two sub-tags. One is a 32×16 SRAM-based sub-tag. The other is a 32×8 CAM-based sub-tag. The high order tag bits are stored in the SRAM sub-tag, which is used to fulfill the first function together with the CAM-based sub-tag. The low order tag bits are stored in CAM-based sub-tag to fulfill the function as a decoder and exploit the replacement policy. During a cache miss, for the most part, the CAM-based sub-tag does not match the desired address; therefore, we can still take full advantage of the replacement policy to find an optimal victim. When the misses occur in the SRAM sub-tag, then we cannot choose an optimum victim from the 32 cache lines. The detailed operation of the proposed LPHAC is shown in the following section. It is apparent that a wider CAM-based sub-tag catches more misses but consumes more power. We determined the optimal CAM-based sub-tag width through experiments.

4.3 Operation of the LPHAC

We use a simple example to show how the proposed LPHAC works. For an address sequence of 0, 1, 3, 7… 0, 1, 3, 7, the operation of the proposed LPHAC is as follows:

First, during the cache's cold start, the contents of both the SRAM sub-tag and the CAM sub-tag are invalid and updated using the desired address. For addresses whose tag bits corresponding to the CAM-based sub-tag are different, such as addresses 0, 1, 3, and 7, the victim is chosen using the replacement policy (least recently used replacement is assumed).

Second, the LPHAC exhibits a miss, but the CAM-based sub-tag exhibits a hit. For example, this happens when address 9 is accessed after the aforementioned address sequence. The CAM-based sub-tag is "001" for address 9 (1001). From Figure 3 (b), the corresponding CAM-based sub-tags are "000", "001", "011", and "111". Since the CAM-based sub-tag of address 9 (1001) is "001", the LPHAC has a CAM sub-tag hit.

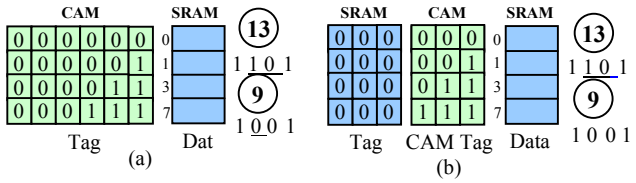

Figure 3: (a) A traditional HAC. (b) The proposed LPHAC

Since only one cache block is activated during an access, the address 9 must replace the address 1. In this situation, the LPHAC cannot choose a better victim based on the access history. If the LPHAC replaces other addresses with address 9, then the address 1 must be evicted as well, to maintain unique address decoding. This definitely impacts the hit rate inadvertently and should be avoided.

Finally, the CAM-based sub-tag exhibits a miss. This happens when address 13 is accessed, since the CAM sub-tag of address 13 (1101) is "101", which is different from the contents stored in the CAM sub-tags, which are "000", "001", "011", and "111". None of the CAM tags match, and no cache line is activated. The victim for address 13 can be chosen from any of the cache lines based on the replacement policy.

5. Experiments

5.1 Experimental Methodology

We use miss rate as the primary metric to measure the effectiveness of the LPHAC. We configure the SimpleScalar [1] as a single-issue in-order processor to collect the miss rate. We determine the CAM sub-tag width through experimentation. The baseline processor is configured with level-one caches of 32-way at size of 8kB with a line size of 32 bytes for both the instruction and data caches. We ran 10 MediaBench and all 26 SPEC2K benchmarks using the SimpleScalar tool set. The SPEC2K benchmarks were fast-forwarded for two billion instructions and executed for 500 million instructions afterwards using reference inputs. For the data cache, all results are reported. For the instruction cache, the results of benchmarks whose miss rates are less than 0.01% are not reported (to save space in the plot), since further reducing the miss rate may not be important for these benchmarks. These benchmarks include *art*, *bzip*, *facerec*, *galgel*, *lucas*, *mcf*, *mgrid*, *swim*, and *vpr*.

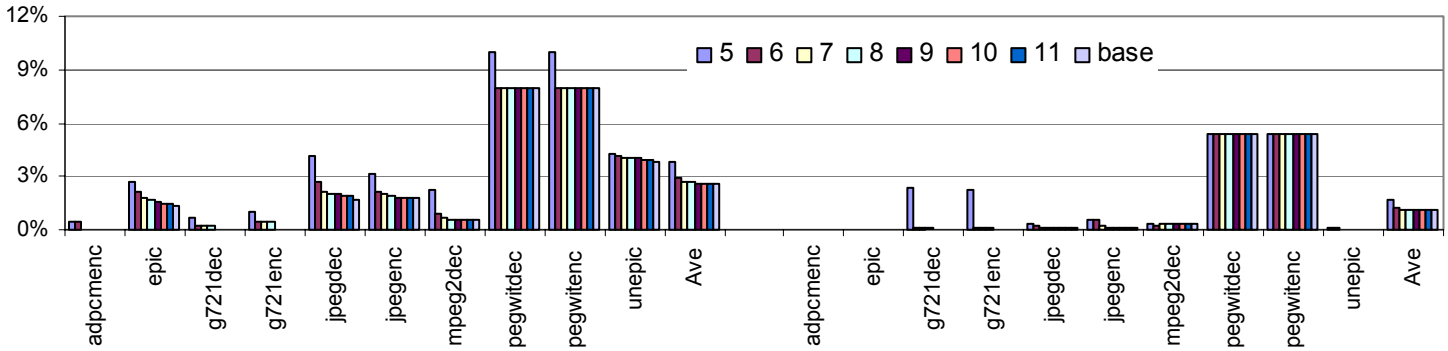

Figure 4: Data (left) and instruction (right) cache miss rates of MediaBench benchmarks. The number from 5 to 10 represents the width of the CAM sub-tag.

5.2 Experimental Results

Figure 4 shows the miss rates of both data and instruction caches for MediaBench benchmarks. Figure 6 and Figure 7 show the miss rates of instruction and data caches of SPEC2K benchmarks, respectively. The bar with "base" represents the miss rate of a traditional 8kB HAC, while the other bars with numbers from 5 to 11 represent the miss rates of the proposed LPHAC with a CAM-based sub-tag width from five to 11 bits. The first observation we made is the miss rate reduction is diminishing when the CAM-based sub-tag width increases higher than eight bits for both instruction and data caches. This means the traditional HAC design overuses the costly CAM. Therefore, we chose eight bits for the CAM-based sub-tag. The corresponding IPC is 0.1% less than the baseline, since some benchmarks, e.g. *vortex*, has a higher miss rate than the baseline.

The second observation we made is that for some applications, such as *perlbmk*, *fma3d*, and *applu*, the lowest miss rate does not occur at the base situation. This is because the least recent used (LRU) replacement policy is not the optimal policy. Recall that the LPHAC has a lower number of CAM cells in the sub-tag, which makes the real policy implemented for the proposed LPHAC different from the traditional LRU. For these benchmarks, the modified LRU policy at certain CAM sub-tag widths is superior to the traditional LRU with full tags implemented using CAM. On the other hand, benchmark *lucas* exhibits the worst miss rate at a CAM-based sub-tag width of 11 bits among all the subtag widths simulated for data cache.

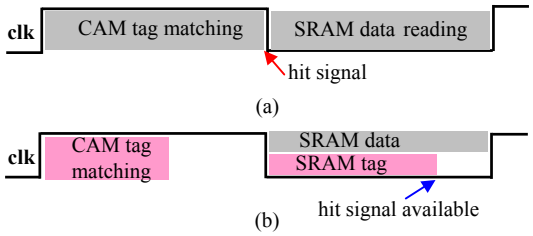

Figure 5: Timing analysis, (a) Traditional HAC; (b) The proposed LPHAC.

6. Analysis

6.1 Timing Analysis

Figure 5 (a) shows the timings of a traditional HAC. The access to a traditional HAC consists of two phases. In the first half-clock cycle, the subbank decoder activates one subbank based on the desired address, and then the tag addresses are fed to the CAM tag store. Tag comparison is finished before the end of the first half-clock cycle. Desired data is read out from the SRAM data stored in the second half of the clock cycle. The hit signal is generated through ORing the 32 match signals from the CAM tag.

For the proposed LPHAC, the CAM sub-tag comparison proceeds faster than the original design, since a 32×8 CAM is 28% faster than a 32×24 CAM based on our HSPICE simulation.

6.2 Area Savings

The area saving comes from the fact that less CAM cells are used in the LPHAC. The area of the CAM cell is 25% larger than the SRAM cell used by data and tag memory. The total storage reduction is calculated as 1.6% of the total cache storage area.

6.3 Power Reduction

The power consumption per cache access is reduced since there are less CAM cells for tag store. We measure the corresponding power reductions using HSPICE simulation of

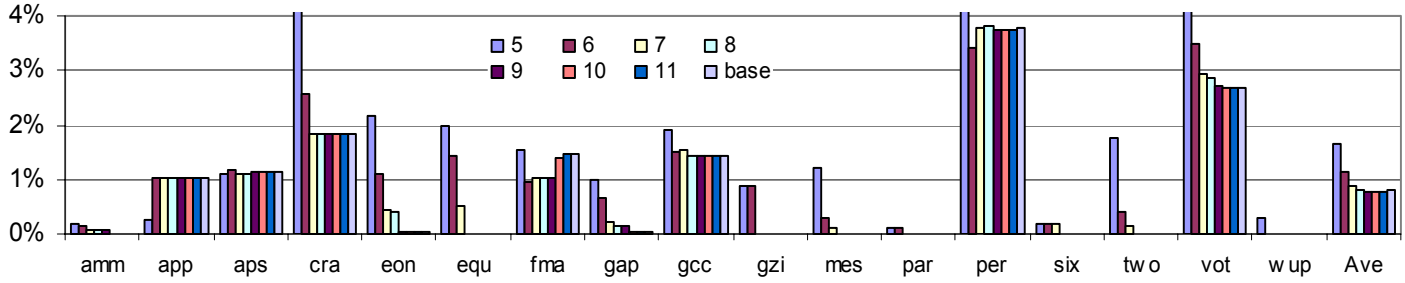

Figure 6: Instruction miss rate of selected Spec2K Benchmarks. *base* stands for the miss rate of a traditional 32-way cache at size of 8kB. The number from five to 11 represents the width of CAM sub-tag.

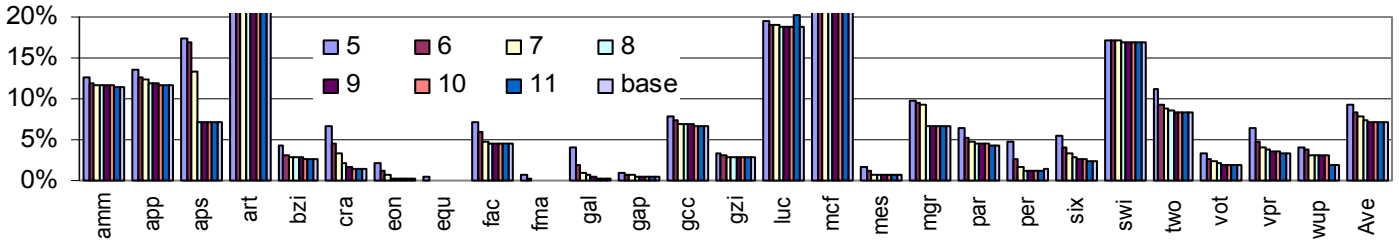

Figure 7: Data cache miss rate of the whole 26 SPEC2K benchmarks. *base* stands for the miss rate of a traditional 32-way cache at size of 8kB. The number from 5 to 11 represents the width of CAM sub-tag.

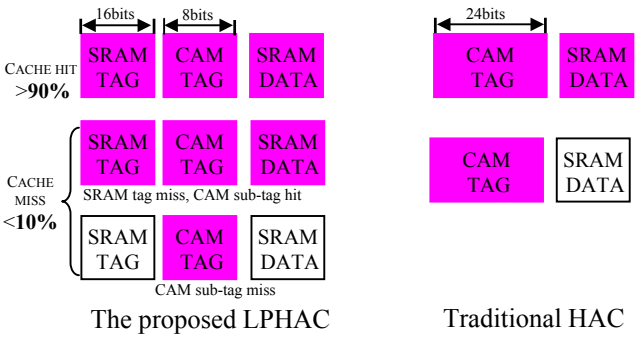

Figure 8: Power dissipation of the proposed LPHAC and traditional HAC during cache accesses.

both the traditional HAC and the proposed LPHAC using 0.18um technology. Figure 8 shows the power consumptions of the proposed LPHAC and traditional HAC during cache hits and misses. During a cache hit, the power per cache access consumption of the proposed LPHAC is reduced by 27% due to the reduction of the CAM tag from 24 bits to 8 bits.

We must point out two subtle situations during a cache miss. In a traditional HAC, no SRAM data will be accessed during a cache miss. However, the situations are different in the proposed LPHAC during a cache miss as shown in Figure 8. One situation is that the cache miss happens in the SRAM sub-tag instead of the CAM sub-tag. Both the SRAM sub-tag and the SRAM data are accessed, since the cache miss can only be determined after looking up the SRAM sub-tag. Under this situation, the proposed LPHAC consumes around 46% more power than a traditional HAC on a cache miss.

Fortunately, this situation happens very infrequently, since most misses occur in the CAM sub-tag. The percentage of the CAM sub-tag hits during cache misses is shown in Figure 9, Figure 10, and Figure 11 for Mediabench and SPEC2k. For Mediabench, the CAM sub-tag hit accounts for 6.5% and 1% of the total cache misses for data and instruction cache, respectively. For SPEC2K benchmarks, the CAM sub-tag hit accounts for 13.8% and 17% of the total cache misses for data and instruction cache, respectively. The cache miss rates on average, as shown in Figure 4, Figure 6, and Figure 7, are less than 1% and 7% for instruction and data cache, respectively. Therefore, the extra power consumed due to CAM sub-tag hits during a cache miss is very limited.

The other situation occurs when the cache miss happens on the CAM sub-tag, and neither SRAM data nor SRAM sub-tag will be read, thus reducing power by 60% compared to the traditional HAC on a cache miss. For Mediabench, the CAM sub-tag misses account for 93.5% and 99% of the total cache misses for data and instruction cache, respectively. For SPEC2K benchmarks, CAM sub-tag misses accounts for 86.2% and 83% of the total cache misses for data and instruction cache, respectively.

Some benchmarks, however, have a very high CAM sub-tag hit rate during cache misses, such as benchmarks *ammp* and *gap* for instruction caches and benchmarks *fma3d* and *galgel* for data cache. The CAM sub-tag hit rates for these benchmarks are higher than 50%. Figure 14 shows the miss rate of benchmark *ammp* at associativities of 1-, 2-, 4-, 8-, and 32-way for a traditional cache and at CAM sub-tag width from 5 to 10 for the proposed LPHAC. The CAM sub-tag hit rate remains 100% till CAM sub-tag is 9-bit and is reduced to zero when the CAM sub-tag is 10-bit. However, the miss rate reduction of the proposed LPHAC at eight bits is already not important. Therefore, eight bits are enough for the CAM sub-tag to reduce the overhead.

6.4 Energy Evaluations

In this section, we compare the energy saving of the proposed LPHAC with a traditional HAC. There are two main components that result in power dissipation in CMOS circuits, namely static power dissipation due to leakage current and

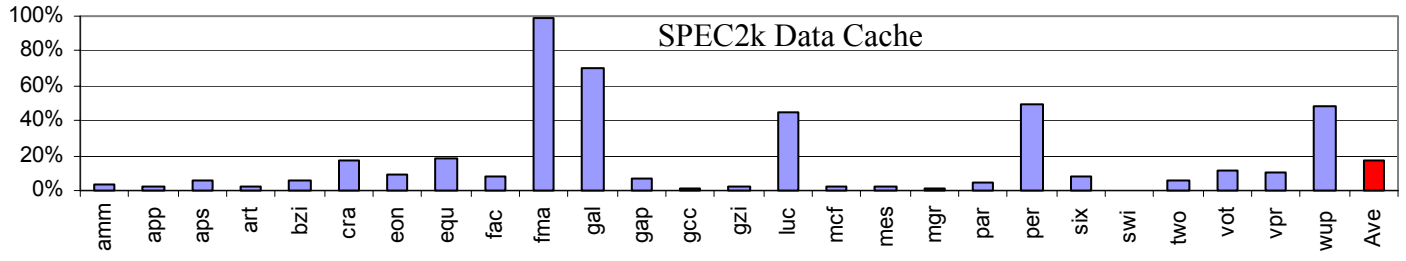

Figure 9: CAM sub-tag hit rates during data cache misses for SPEC2K benchmarks.

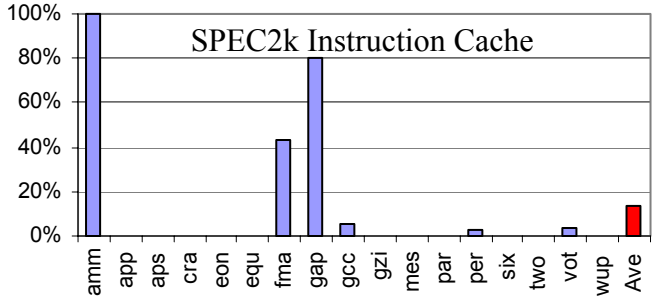

Figure 10: CAM sub-tag hit rates of the instruction cache during cache misses for SPEC 2K benchmarks.

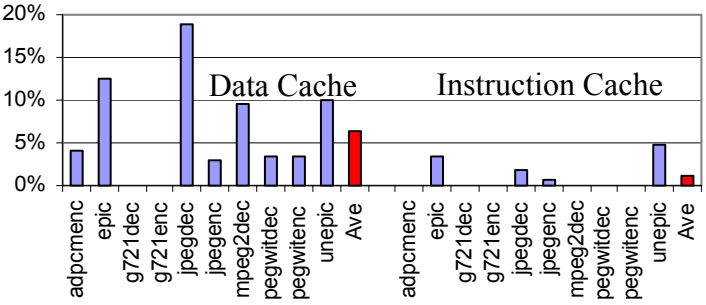

Figure 11: CAM sub-tag hit rates of both the data and instruction caches during cache misses for Mediabench benchmarks.

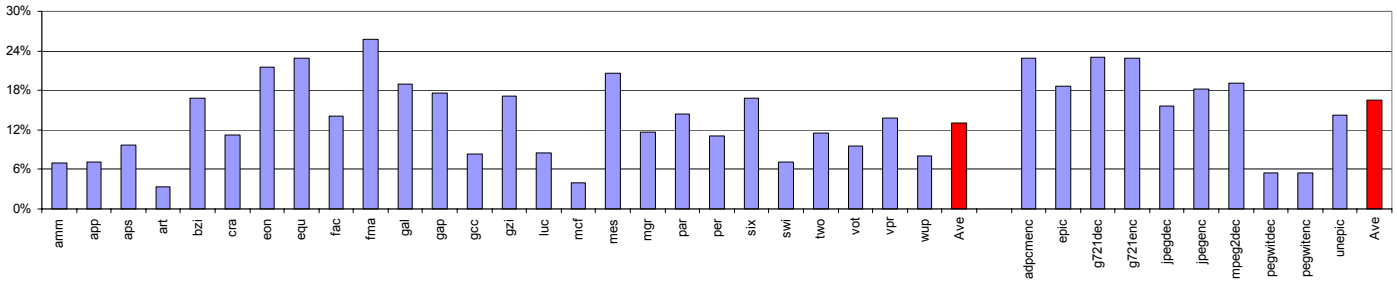

Figure 12: Energy reductions of the proposed LPHAC compared to the tradition HAC for SPEC2K and Mediabench benchmarks**.**

E_mem = cache_access * E_cache_access + cache_miss * E_misses

E_misses=E_next_level_mem+E_cache_block_refill

Figure 13: Equations for energy evaluation.

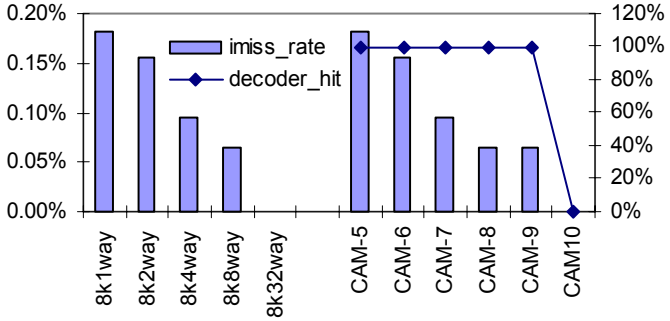

Figure 14: Instruction cache miss rates of benchmark *ammp* at varied associativities for a traditional cache and the proposed LPHAC at CAM subtag widths from 5 to 10.

dynamic power dissipation due to logic switching current and the charging and discharging of the load capacitance. The static energy is proportional to the cache size and execution time of an application. The data store of both LPHAC and the traditional HAC is identical, and the tag store of LPHAC is smaller than that of baseline; however, we do claim this reduction. The execution time or IPC of all the benchmarks remain almost unchanged, since the miss rate of the proposed LPHAC is almost the same with the traditional HAC. Therefore, we consider only dynamic energy in our evaluation. We evaluate the memory related energy consumption (E_mem) including on-chip caches and off-chip memory. Figure 13 lists the equations for computing the total memory related energy consumption.

The *italic* terms are those we obtain through measurements or simulations. We compute *cache_access*, *cache_miss*, and *cycles* by running SimpleScalar simulations for each cache configuration. We compute *E_cache_access* and *E_cache_block_refill* using Cacti 3.2 and the HSPICE simulation for both LPHAC and the baseline.

The *E_next_level_mem* includes the energy of accessing off-chip memory. Using a methodology similar to [12], we evaluate the energy of accessing off-chip memory as 50 times larger than the on-chip cache.

Figure 12 shows the energy of the LPHAC normalized to the baseline. On average, the LPHAC consumes 13% and 16.5% less energy than the baseline for Mediabench and SPEC2K, respectively. The greatest energy reduction is seen in *fm3d*, where the energy is reduced by 25.7%. This high energy reduction is because the instruction miss rate of benchmark *fm3d* is lower than that of the baseline as explained in Section 5.2. The miss rate reduction when using 8-bit CAM based sub-tag is higher than the baseline where 24-bit CAM based tag is used.

On average, the energy reduction for Mediabench and SPEC2K are 13% and 16.5%, respectively.

7. Conclusion

We propose a low power design for high associative caches. The proposed LPHAC employs an 8-bit instead of 24-bit CAM so it consumes less per access power and area than a traditional HAC while exhibiting a faster access time. The memory accessed related energy reduction can be as high as 25.7% with averages of 13% and 16.5% for Mediabench and SPEC2K, respectively. Compared with other related low power highly associative caches, the proposed LPHAC incurs no hardware or software overhead.

References

[1] D. Burger and T.M. Austin, “The SimpleScalar Tool Set, Version 2.0,” Univ. of Wisconsin-Madison Computer Sciences Dept. Technical Report #1342, June 1997.

[2] A. Efthymio and J.Garside,”An Adaptive Serial-Parallel CAM Architecture for Low-Power Cache Blocks.” In Proc. of ISLPED, 2002.

[3] Intel. Intel XScale Microarchitecture, 2001.

[4] C. Lee, M. Potkonjak and W. Mangione-Smith, “MediaBench: A Tool for Evaluating and Synthesizing Multimedia and Communications Systems,” Int. Symp. on Microarchitecture, 1997.

[5] A Ma, M Zhang, K Asanovic,” Way memorization to reduce fetch energy in instruction caches,” ISCA Workshop on CED, 2001.

[6] J. Montagnaro and et al.,”A 160–MHz, 32–b, 0.5–w CMOS RISC microprocessor. IEEE JSSC, 31(11):1703–1714, Nov. 1996.

[7] K. Pagiamtzis and A. Sheikholeslami, A Low-Power Content-Addressable Memory (CAM) Using Pipelined Hierarchical Search Scheme, IEEE Journal of Solid-State Circuits, Sep. 2004.

[8] G. Reinmann and N.P. Jouppi. CACTI2.0: An Integrated Cache Timing and Power Model, 1999. COMPAQ western Research Lab.

[9] S. Santhanam, et. al. “A Low-Cost, 300-MHz, RISC CPU with Attached Media Processor,” IEEE Journal of Solid-State Circuit, Vol. 33, 1998.

[10] http://www.specbench.org/osg/cpu2000/

[11] A Veidenbaum and D Nicolaescu, ”Low Energy, Highly-Associative Cache Design for Embedded Processors,” IEEE ICCD, 2004.

[12] C. Zhang, F. Vahid, and W. Najjar, “A Highly-Configurable Cache Architecture for Embedded Systems,” In proceedings of International Symposium on Computer Architecture, 2003, San Diego.

[13] C. Zhang, F. Vahid, J. Yang and W. Najjar,”A Way-Halting Cache for Low-Energy High-Performance Systems,” ISLPED 2004.

[14] C. Zhang, “Balanced-Cache: Reducing Conflict Misses of Direct-Mapped Caches through Programmable Decoders,” In proceedings of International Symposium on Computer Architecture, 2006, Boston.