Low Energy, Highly–Associative Cache Design for Embedded Processors

Alex Veidenbaum Dan Nicolaescu
Computer Science Department
University of California, Irvine
{alexv,dann}@ics.uci.edu

Abstract

Many embedded processors use highly associative data caches implemented using a CAM-based tag search. When high-associativity is desirable, CAM designs can offer performance advantages due to fast associative search. However, CAMs are not energy efficient. This paper describes a CAM-based cache design which uses prediction to reduce energy consumption. A last used prediction is shown to achieve an 86% prediction accuracy, on average. A new design integrating such predictor in the CAM tag store is described. A 30% average D-cache energy reduction is demonstrated for the MiBench programs with little additional hardware or impact on processor performance. Even better results can be achieved with another predictor design which increases prediction accuracy. Significant static energy reduction is also possible using this approach for the RAM data store.

1. Introduction

Embedded Processor performance is highly dependent on cache performance, which in turn depends on a number of cache parameters. Associativity is one such parameter, with higher associativity leading to higher performance, even though higher associativity increases the cache access latency. Not surprisingly, several embedded processors have implemented highly associative caches [5],[10],[8].

Cache energy consumption is another major problem in embedded processors [3] and needs to be further reduced. As performance requirements continue to grow, more energy efficient direct mapped or low-associativity designs are less desirable. 32–way set associative caches have already been implemented. Such highly associative caches used CAM-based tag storage for fast search. While fast, CAMs are not energy efficient.

The energy consumption of such highly associative, CAM–based caches is significantly smaller than that of RAM-based designs of the same associativity and speed [3]. The energy consumption difference between the two types of tag store designs increases with the increased associativity [14]. However, further energy savings are still desirable. This paper proposes an approach that can achieve this by reducing both static and dynamic energy requirements of CAM–based designs with minimal impact on performance. This is achieved by predicting which cache lines are likely to be accessed and applying energy reduction techniques to the other lines in the cache.

This work makes the following contributions. A new CAM-based cache design integrating a predictor is proposed, which significantly reduces the energy dissipated by match lines. An even higher accuracy can be achieved with an advanced version of the last-use predictor. Improving accuracy is important since misprediction is costly in terms of both latency and energy. It is also shown that it is possible in some cases to improve cache latency when prediction is correct.

Finally, the prediction mechanism can also be used to reduce static energy consumption. It can utilize existing hardware techniques, such as drowsy cache [4], to put cache lines in a reduced-energy state based on predicted use.

2. Operation of CAM-based caches

The cache operation of a CAM-based set-associative design differs from that of a RAM-based design. First, the cache is partitioned in such a way that a set is mapped to a single CAM/RAM block. There are M such blocks in the cache, as shown in Fig. 1, where
$M = Cache\_size/(Associativity * Line\_size)$.
Log(M) bits of a CPU address are used to select one CAM/RAM block, which uses the CAM as a tag store and the RAM as a data store. Each such block is basically a small, fully associative sub-cache.

Once the sub-cache has been selected by the corresponding part of the address, the CAM tag store and the data RAM store operate sequentially. First, a CAM tag lookup checks if a line is present in the sub–cache, generating a $Match_i$ signal for each tag (see Fig. 2). This is done in parallel for each tag in the CAM. Only a single $Match_i$ signal can become a ”1” on a hit.

The match lines serve as inputs to wordline drivers for each cache line in the RAM part, eliminating the need for address decoding. Once a wordline is activated, the desired number of columns is selected, and written or latched for output.

The CAM lookup is very energy-intensive as each $Match_i$ line is first precharged and then discharged for every tag that did not match a search address. This is estimated to dissipate approximately one half of all the CAM lookup energy. The other half is dissipated driving the search address to all comparators. The two components are approximately equal for small CAM blocks of 32 to 64 entries. The subsequent RAM access is also energy intensive and requires approximately the same amount of energy as the CAM lookup.

3. Which lines are likely to be accessed?

Many techniques have been proposed to predict future line usage in I- and D-caches. They can be based on past usage or on future address prediction. Past usage is a more desirable approach in the context of energy management. Due to locality, a cache line is likely to be accessed multiple times. Therefore, once a line has been accessed it is likely to be accessed again. Lines that have not been accessed recently, on the other hand, are less likely to be accessed. Keeping track of this information can help manage the access energy. As far as this work is concerned, the goal is to identify a small number of such lines and spend the energy on accessing these lines while saving energy on other lines.

Identifying recently accessed lines requires hardware resources, such as a cache of recently used line numbers [11; 13] or RAM space [2; 9]. These hardware structures introduce a delay for accessing the line information. This additional delay may not acceptable in an embedded processor. It is also desirable to further limit the number of lines tracked.

A *last use* predictor has been shown to be successful in number areas: line/set prediction[9], way prediction [7], branch prediction, etc. In the set-associative cache organization considered in this paper, the last line used in a set is likely to also be the next one to be used. Therefore, last use prediction is chosen for this work. A new design, proposed here, integrates such "last use" predictor into a CAM–based tag store.

Fig.4 shows the accuracy of the *last use* predictor. It demonstrates a very high degree of prediction accuracy, even though several programs have a significantly lower accuracy. Still, on average the accuracy is sufficient for this use, as will be shown in Sec. 5.2. It is possible to further improve the accuracy improved using a *multiple last use* prediction scheme.

The next section presents the cache design using the *last use* information to reduce dynamic energy consumption.

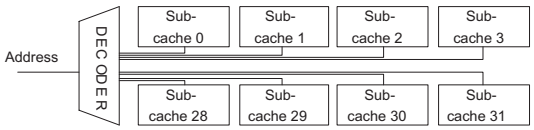

Figure 1: Address decoder and fully associative sub–cache blocks

4. The proposed cache design

The *last use* (LU) mechanism mentioned above is deployed to predict which line in the fully associative sub-cache is most likely to be accessed next, for each sub-cache. The new design shown in Fig.3 adds a latch, LU_i, which maintains the last use information between accesses to the sub-cache. This information is used to disable precharge for line(s) that have not been recently accessed. Thus tag compare is only performed on the last tag used. Energy is saved if the prediction is correct and the access is a hit. On a "last use" miss, a full precharge cycle is run, followed by a full CAM tag search.

It is assumed that the RAM access waits for the CAM to complete the tag compare and to verify the prediction. Since there is only one possible "last use" line, the Miss in this case implies an LU miss. The Miss information (32-input OR) is first used to initiate a full CAM precharge and CAM access. It is also used as a stall signal to the (in-order) processor pipeline. Finally, a true Miss is generated if the full CAM access also indicates a miss.

What are the dynamic energy savings possible from the proposed design? The upper bound is 31/32nd of the comparison energy for a 32-entry CAM with no misprediction. Recall that the comparison accounts for approximately one half of the total CAM energy consumption. Thus the maximum savings are 48%. Mispredictions will lead to lower savings.

In the design shown, the RAM is only accessed after the last use prediction is verified. This is conservative and can be avoided if latency were very important. In this case the RAM access will be started at the same time as the tag access and before the verifying the last use prediction. In case of mispredicitons the RAM access will be incorrect and will have to be repeated.

The new design can also be modified to include static (leakage) energy management. The LU information can be utilized to keep lines in a low-leakage mode. For instance, assume that the drowsy cache cell design [4] is used, which can put an entire cache line in a "drowsy", low-leakage mode. In this case each LU line can be kept in the normal mode while all lines with LU=0 can be put in the drowsy mode. Transition from drowsy to normal state will require an extra clock cycle and has an energy cost. However, it allows 31/32 of all the lines in the cache (including their tags) to be in a low-leakage state.

The cache design described in this section (without speculative RAM access) is experimentally evaluated next.

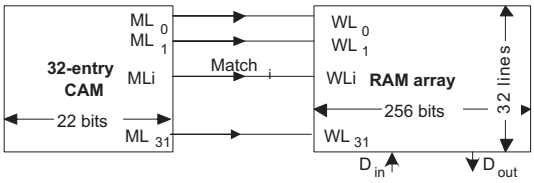

Figure 2: A fully associative sub–cache

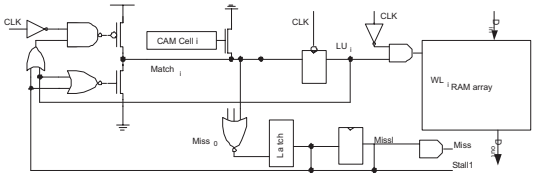

Figure 3: New sub–cache design

5. Experimental evaluation

The proposed cache design was evaluated using an Xscale-like processor model. It is a single issue, in-order CPU, with a bimodal branch predictor. The predictor and the branch target buffer has 128 entries. The processor has one complex arithmetic unit for integer multiplication and division. The pipeline length is 7 cycles.

The simulated processor uses 32KB, 32–way set associative data and instruction L1 caches, both with 32 byte lines. The L1 cache access latency is 2 cycles, one cycle for CAM tag lookup and RAM access, and one cycle for data transfer to the CPU core. An LU misprediction is detected at the end of the first cycle and a 2 cycle access is repeated, resulting in a 3 cycle mispredicted access latency.

The fully associative instruction and data TLBs have 32 entries. The system does not contain an L2 cache. Main memory access latency used was 80 cycles.

The proposed design was simulated using SimpleScalar-3.0d [1] simulator which was augmented with a cache model using an LU_1 predictor.

A 0.09 micron implementation technology is assumed resulting in a clock frequency of 1GHz. The dynamic energy consumption model for the new cache was obtained using a modified CACTI 3.2 [12] model. The main changes were to the sense amplifier energy for RAM access (overestimated by CACTI) and to the CAM address drivers (underestimated by CACTI). The CAM compare energy was assumed to be 1/32nd of the baseline when the LU_1 predictor was used. The RAM model assumed a wide line with column multiplexing resulting in a single 32b word access. The additional logic of the proposed cache is minimal and is assumed to consume no energy.

Evaluating static energy impact is more difficult since it requires an accurate model for leakage current. Leakage energy and transition energy per bit from [4] were scaled to 90nm and used in our evaluation. The average static energy consumption is approximately 30% of the dynamic cache energy consumption to the base case.

5.1. Benchmarks

MiBench[6] is a publicly available benchmark suite designed to be representative for several embedded system domains. The benchmarks are divided in six categories targeting different parts of the embedded systems market. The suites are: Automotive and Industrial Control (basicmath, bitcount, susan (edges, corners and smoothing)), Consumer Devices (jpeg encode and decode, lame, tiff2bw, tiff2rgba, tiffdither, tiffmedian, typeset), Office Automation (ghostscript, ispell, stringsearch), Networking (dijkstra, patricia), Security (blowfish encode and decode, pgp sign and verify, rijndael encode and decode, sha) and Telecommunications (CRC32, FFT direct and inverse, adpcm encode and decode, gsm encode and decode). All the benchmarks were compiled with the -O3 compiler flag and were simulated to completion using the “large” input set.

5.2. Results

Results in Fig. 4 show that using the LU predictor achieves an average 30% reduction in the data cache dynamic energy consumption. This is below the optimal 48% savings possible and is explained by an average 15% LU misprediction rate. Individual benchmarks have misprediction rates as low as 0.01% and as high as 38%, as shown in Fig.4. However, even the worst misprediction rate still leads to 20% cache energy reduction.

The high misprediction rate of some benchmarks is a problem because it affects the execution time as well as reducing the energy savings in the cache. Recall that the cache access is two cycles, with misprediction detected in the first cycle. Thus the effective cache access latency in case of misprediction is 3 cycles. The additional delay results in a CPI increase of up to 3% (worst case). This is an acceptable trade-off between energy reduction and small performance loss, since the energy-delay product is improved, on average, by 30%.

The high misprediction rate observed in several benchmarks can be overcome with a generalized LU predictor called *n–th order LU* predictor. The n–th order LU predictor, LU_n, keeps track of the last n lines accessed. The cache design proposed here can be easily extended to use the LU_n predictor for $n > 1$. This will increase the CAM energy consumption but will decrease the misprediction rate and the associated energy loss and delay. The detailed evaluation of the LU_n predictors and their impact on energy consumption is beyond the scope of this paper. The highest impact is on benchmarks that had the worst misprediction rate with the LU_1 predictor. For instance, a 20% prediction accuracy improvement can be achieved for the ”rijndael” benchmarks using an LU_4 predictor. The average prediction accuracy for LU_4 is 97%, a 10% improvement over the LU_1 predictor.

The proposed design is also effective for reducing the

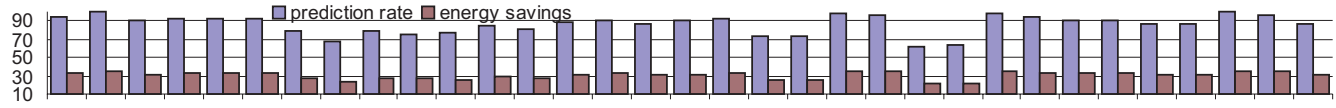

Figure 4: Prediction accuracy of the LU predictor and the corresponding dynamic energy savinngs

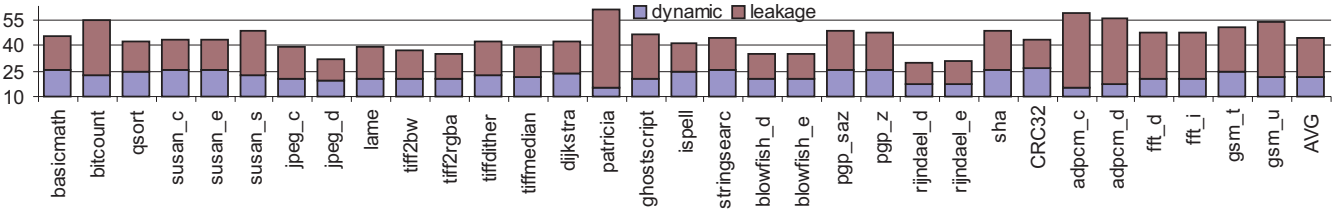

Figure 5: Static and dynamic energy savings, including drowsy–normal transition energy

leakage current and energy. The high LU prediction accuracy implies that most of the time the majority of lines in the RAM array will be in the low-energy state. Fig. 5 shows combined dynamic and leakage energy savings for the proposed design. The average savings for the entire benchmark suite are 44%., with approximately half of the reduction from the static energy. The figure includes contributions from transition energy (to activate the drowsy lines) and delay (recall that the misprediction latency in this case is 4 cycles).

6. Conclusions

The design of an energy efficient highly associative cache has been presented. It integrates a last-use predictor with a CAM-based tag store to significantly reduce the dynamic energy consumption of the cache. Experimental evaluation of a 32-way set-associative data cache of an embedded processor demonstrated an average 30% dynamic energy reduction for MiBench codes. The new cache design relies on prediction and thus can incur a loss of performance due to increased cache access latency in the case of misprediction. It is shown that the maximum CPI loss is only 3% and the average energy-delay product reduction is nearly 30%.

The proposed design can be applied to the instruction cache with no change. The I-cache LU prediction rate is even higher due to locality of instruction accesses and will lead to even higher savings.

Generalized LU predictors were also briefly described. Keeping track of two to four most recently used lines instead of just the last use significantly increase prediction accuracy. For benchmarks with the worst misprediction rate, rijndael's, a 10 to 20% improvement is possible.

Finally, the proposed design can also be used to reduce the static (leakage) energy consumption. The LU information can be used in conjunction with a number of reduced leakage RAM cell designs, such as the drowsy cache cell, to keep the lines not likely to be used in a reduced energy state. This leads to an additional 20% average total cache energy savings from reduced leakage current using drowsy RAM cells in the 90nm technology used in this paper.

References

[1] D. Burger and T. M. Austin. The SimpleScalar tool set. Technical Report TR-97-1342, University of Wisconsin, 1997.

[2] B. Calder and D. Grunwald. Next cache line and set prediction. In *Proceedings ISCA*, pages 287–296, 1995.

[3] L. T. Clark and et al. An embedded 32b microprocessor core for low-power and high-performance applications. *IEEE JSSC*, 36(11):1599–1608, Nov. 2001.

[4] K. Flautner and et all. Drowsy caches: simple techniques for reducing leakage power. In *ISCA*, pages 148–157, 2002.

[5] S. Furber and et al. ARM3 - 32b RISC processor with 4kbyte on-chip cache. In *Proceedings VLSI*, pages 35–44, 1989.

[6] M. R. Guthaus and et all. Mibench: A free, commercially representative embedded benchmark suite. In *IEEE WWC*, pages 83–94, 2001.

[7] K. Inoue, T. Ishihara, and K. Murakami. Way-predicting set-associative cache for high performance and low energy consumption. In *ISLPED*, pages 273–275, 1999.

[8] Intel. *Intel XScale Microarchitecture*, 2001.

[9] R. E. Kessler. The Alpha 21264 microprocessor. *IEEE Micro*, 19(2):24–36, Mar./Apr. 1999.

[10] J. Montagnaro and et al. A 160–mhz, 32–b, 0.5–w cmos risc microprocessor. *IEEE JSSC*, 31(11):1703–1714, Nov. 1996.

[11] D. Nicolaescu, A. Veidenbaum, and A. Nicolau. Reducing power consumption for high-associativity data caches in embedded processors. In *DATE2003 Proceedings*, 2003.

[12] P. Shivakumar and N. P. Jouppi. Cacti 3.0: An integrated cache timing and power model. Technical report, DEC, 2001.

[13] E. Witchel and et all. Direct addressed caches for reduced power consumption. In *MICRO-34*, 2001.

[14] M. Zhang and K. Asanovic. Highly-associative caches for low-power processors. In *Kool Chips Workshop*, 2000.