Modeling and Analysis of The Difference-Bit Cache

Ashutosh Kulkarni, Navin Chander, Soumya Pillai and Lizy John

Department of Electrical and Computer Engineering
The University of Texas at Austin
Austin, TX 78712
{akulkarn,chander,pillai,ljohn}@ece.utexas.edu

Abstract

Advances in VLSI technology and processor architectures have resulted in a tremendous increase in processor speeds and memory capacities. However memory latencies have failed to improve as rapidly, making memory systems the performance bottlenecks in most high performance processor architectures. Caching is a time-tested mechanism to solve this speed disparity. Among the different cache mapping strategies, direct mapping is the only configuration where the critical path is merely the time required to access a RAM. Although direct mapped caches are preferable considering hit-access times, they have poor hit ratios compared to associative caches. The difference-bit cache proposed by Juan, Lang and Navarro [1], is functionally equivalent to a two-way set-associative cache but tries to achieve an access time smaller than that of a conventional two-way set-associative cache and close to that of a direct-mapped cache. We modeled and analyzed the difference-bit cache to prove the hypothesis of its small access time. We have also tried to prove that the access time advantage of the difference-bit cache improves over the conventional two-way set-associative cache with an increase in the cache size. Finally we have tried to analyze the trade-off involved in applying these techniques to a higher associativity cache.

Keywords: Cache memory, critical path, hit access time, cache mapping strategies[*]

1 Introduction

A common feature of all modern microprocessor architectures, be it superscalar processors, VLIW processors or multiscalar processors [6], is the ability to issue and execute multiple instructions per cycle. It is therefore of paramount importance that the memory systems in these computers provide the bandwidth required to support these architectural advances. Multiple cache levels are usually used for this so that small, high speed, static random access memory devices can feed the high performance microprocessors to avoid the high latencies of dynamic random access main memories. Since the direct-mapped cache provides the best access time, it is usually used as the first-level cache with set-associative cache schemes used in the second and sometimes the third levels of the cache hierarchy. The direct-mapped cache however, has the highest miss ratio for most programs [7][8]. Having a cache with the functionality (and hence the miss ratio) of a set-associative cache and the access time of a direct-mapped cache at the first level would most likely give the best overall performance.

Several schemes have been suggested [3][4][5] where the set-associative cache has been modified to achieve an access time close to that of a direct-mapped cache. Most of these use some sort of prediction scheme for selecting the block from within a selected set. If this prediction is correct the access time is just one processor cycle (similar to that of a direct-mapped cache), but if the prediction turns out to be incorrect, the penalty incurred can be about two to four processor cycles. As a result the average access time is still larger than that of the direct-mapped cache. The difference-bit cache proposed in [1] differs from these implementations in that it uses no prediction scheme. Hence there is only one type of hit that can have a latency equal or close to that of a direct-mapped cache depending on the implementation.

In this paper, we present an implementation and analysis of the difference-bit cache. Juan, Lang and Navarro [1] had made their hypothesis about the cache latencies based on the cache access time model presented in [2]. We were able to verify quantitatively, the access time improvements of the difference-bit cache over the conventional set-associative cache. We were also able to prove that the latency improvement of

[*] This work was supported in part by the National Science Foundation under Grant CCR-9796098

the difference-bit cache over the conventional cache increases with an increase in the cache size.

1.1 Related Research

Several proposals have been made to achieve an average access time close to that of a direct-mapped cache from a two-way set-associative cache. A vast majority of them predict the way and select the corresponding word. Later, after the tags have been compared, the correct way is determined and if the prediction has failed, a new selection has to be performed. This results in two types of hits

- Primary hit when the prediction is correct, resulting in a one cycle access time and
- Secondary hit when the prediction fails, resulting in a new selection and hence an access time of two to four cycles.

The MRU cache [4] selects the most recently used block, resulting in a secondary hit access time of two cycles. Though this scheme can be used for all degrees of associativity, the probability of a primary hit decreases with increase in associativity. In the DASC cache [5], prediction is done assuming a direct-mapped cache. If the tag side detects a hit in the other position of the set, the accessed block and the correct block are swapped, resulting in a secondary hit access time of four cycles. In case of a cache miss, the block is written in accordance with the replacement algorithm and then swapped with the block that is accessed in a direct-mapped cache. This scheme too can be used for any degree of associativity. The PAD cache [3] has a tag side divided into two parts, the first of which holds the k least-significant bits of the tags and the other, the remaining bits. The way is predicted by comparing the tags of the first part. In case of more than one hit in this part, any of the ways is accessed (depending on the algorithm used, for example, the most recently used) while the other part of the tags are compared to determine whether the correct way was predicted.

1.2 Contributions

Our work is an implementation and analysis of the difference-bit cache [1] that tries to achieve access time comparable to that of a direct-mapped cache in a two-way set associative cache without using any form of prediction. For their analysis the authors of the original paper on the difference-bit cache had used a model for cache access times proposed by S. Wilton and N. Jouppi [2]. This model was based on HSpice modeling of various components of on-chip caches. However no actual cache modeling was attempted in [1]. We have modeled the difference-bit cache and compared it quantitatively with direct-mapped and two-way set-associative caches for access times. In Section 2, we describe the difference-bit cache and highlight the difference in its architecture as compared to the direct-mapped and two-way set-associative cache. In section 3 we present a quantitative analysis of the access times of all the three caches as well as an analysis of extending the scheme for higher associativity and in section 4, we conclude with a summary of our results.

2 The Difference-Bit Cache

The optimal (fastest) realization of the data part of the cache memory consists of implementing it as several subarrays, the number and size of which depends on the cache size, technology characteristics and implementation restrictions. For our modeling, the index bits in the direct mapped case were partitioned into two parts, one of which was used to access the block from each subarray, the other being used to select the desired subarray (figure 1).

The two-way set associative cache used the same partitioning of the index bits, with the difference that the signal used to enable the desired subarray (way) was obtained from tag comparisons, instead of directly from the index (in the memory address from the processor). The access time of the two-way set associative cache is larger than that of a direct-mapped cache because the critical path is through the tag part and includes the access of the tags and the comparisons (figure 2).

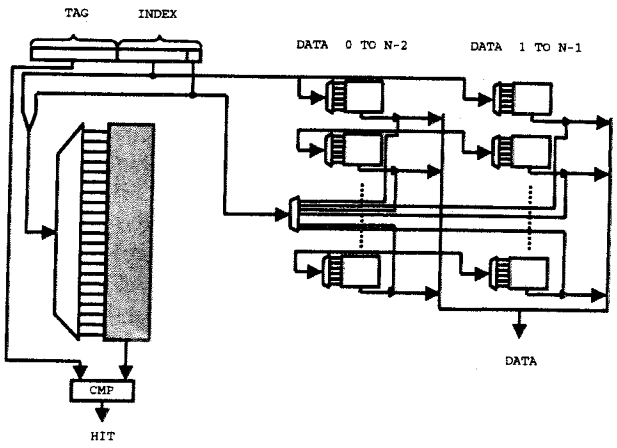

Figure 1: A two-subarray implementation of a direct-mapped Cache

In the case of the direct-mapped cache, the delay of the enable signal to the tri-state gates is smaller than the access time of the data memory part. This causes the time in which the resulting data word is obtained to be equal to the sum of the time in which the data is available at the output of the subarray plus the delay of the tri-state gate. Also, data acquisition and determination of whether the access is a hit, can proceed in parallel since the tag comparator is not in the critical path. Information about whether the access is a hit or not is needed only before the data is actually used (say for storage in a register).

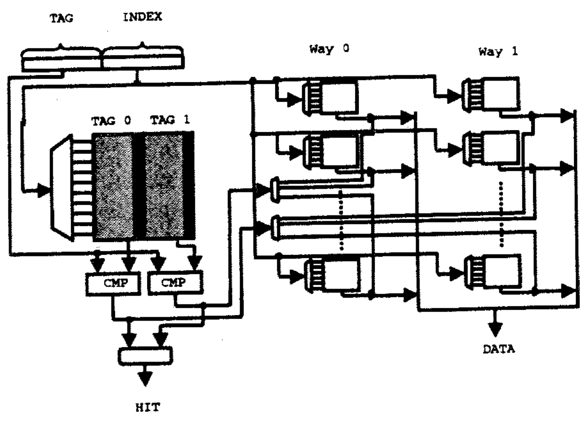

Figure 2: Implementation of a Conventional Two-way Set-associative Cache

To have a set-associative cache with the access time of a direct-mapped cache, it is sufficient that the enable signals of the tri-state gates are obtained with a delay that is smaller than the access time of the data subarray. The explanation given below uses this sufficient condition analysis and illustrates how the difference-bit cache achieves direct-mapped cache access time.

2.1 Architecture of the Difference-Bit Cache

A key observation in the two-way set-associative cache *is that the two tags that correspond to a set differ in at least one bit*. Two new parameters *diff-index* and *diff-value* have been proposed to encode information regarding these differing bits. The *diff-index* is the position in the tag of the least-significant bit in which the two tags differ and *diff-value* is the value of the bit in the tag of way 0 of the set. These diff-index and diff-value are used to determine the enable signal to the tri-state gates (to determine which block to select in the set). For this a new memory array called the *diff-array* (figure 3) is introduced into the cache. The size of this memory is NS x b where NS is the number of sets of the two-way set-associative cache and b depends on the code used to represent the diff-index. If normal binary code is used for the representation, the value of b is a minimum of $\{log_2 t + 1\}$ where t is the number of bits of the tag.

The enable signals of the tri-state gates are obtained as explained below:

1. The corresponding entry of the *diff-Array* is read simultaneously with the data in the data part of the cache and the tags in the tag part (although the tag part is not in the critical path).
2. The *diff-index* obtained is used to point to the corresponding bit of the tag portion in the address.
3. The selected bit in the address and the *diff-value* are compared; if they are equal, way zero of the set is selected whereas if they are different, way one is selected.
4. The way bits are used to drive the enable signal of the tri-state gates that pass the corresponding data block.

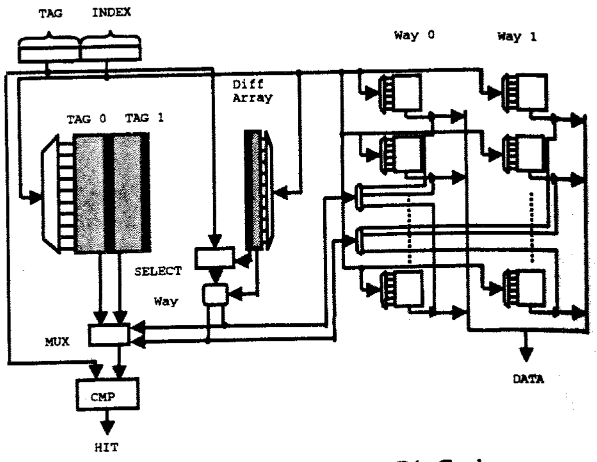

Figure 3: The Difference-Bit Cache

In case of a miss, it is necessary to determine the *diff-index* and *diff-value* for the new data in the cache. This involves a very simple array of XOR gates to compare the bits of the tags and an encoder for the code used to represent the *diff-index*. The replacement algorithm used is the same as for the two-way set-associative cache thus resulting in the same miss-ratio.

3 Modeling and Analysis

3.1 Analysis of Critical Path

In this section, we present a discussion on the critical paths of the cache models to prove the feasibility of the difference-bit cache. This analysis is primarily based on the model analysis presented in [1].

The critical path as seen in figure 3, is given by

$$t_c = \max(t_{data}, t_{enable}) + t_{tri}$$

where t_{data} is the time required to access the data subarray of the cache, t_{enable} is the time for the enable signal to select the way and t_{tri} is the time required to enable the tri-state gates of the data part of the cache. Consequently, the access time corresponds to that of a direct-mapped cache if

$$t_{enable\ of\ difference\ bit\ cache} \leq t_{data}$$

where $t_{enable} = t_{diff} + t_{select} + t_{way}$

and t_{diff} is the *diff* memory access time.

The values of t_{diff} and t_{select} are related and depend on the coding scheme used to represent the *diff-index*. If the code has more bits, t_{diff} is larger since the memory is wider, but tselect is smaller since the decoding is simpler. Now, the *diff-array* is significantly smaller than one subarray of data. This is because the data memory has a width of one line of data (say L bits) whereas the *diff-array* has a width of b where $b \ll L$.

The important assumption for this analysis to hold is that the *access time of a memory subarray is directly dependent on the complexity of its decoder, which in turn is directly proportional to the memory size*. The *diff-array* which has a minimum width of $\{log_2 t + 1\}$ where t is the number of bits of the tag (when binary coding is used for diff-index), is much smaller than a data subarray of a direct-mapped cache. Hence the difference-bit cache should have an access time close to or equal to that of a direct-mapped cache.

3.2 Critical Path results from modeling

We implemented direct-mapped, two-way set-associative and difference-bit cache models using VHDL and Synopsys' Design Analyzer as a synthesis tool. The functional accuracy of our models was verified using the Synopsys VHDL simulator. The description methodology was adopted to enable specification in an efficient manner, giving us greater latitude of freedom while specifying constraints. Schematic build-up and representation was done using Synopsys' Design Analyzer and Design Compiler. Once the schematic was obtained, it was then optimized for timing, and analysis was done for calculating the critical path delays in the cache models. Synopsys is very much the de-facto tool used for logic synthesis and optimization both on the gate level as well as at a higher level of abstraction. The design optimization and analysis features allow for point to point timing delay calculations. Custom libraries built into the tool allow for timing and area analysis on different VLSI technologies.

For our implementation we used the following cache configuration:

Cache size: 64 bytes
Block size: 1 byte

and assumed a main memory of size 1 KB. The results that we obtained are tabulated in table 1.

As indicated in table 1, the time delay of the enable logic of the difference-bit cache is smaller than the access time of a data subarray in the direct-mapped cache. This proves the hypothesis that the difference-bit cache would be able to achieve access times close to that of the direct-mapped cache. In the case of the two-way set-associative cache, we see a larger enable logic delay than the direct-mapped cache.

Direct-Mapped Cache (Data Subarray Delay)	Set-Associative Cache (Enable Signal Delay)	Difference-Bit Cache Enable Signal Delay
14.35 ns	16.71 ns	12.45 ns

Table 1. Results for cache size of 64 bytes, block size of 1 byte and 1 KB main memory

Direct-Mapped Cache (Data Subarray Delay)	Set-Associative Cache (Enable Signal Delay)	Difference-Bit Cache Enable Signal Delay
22.42 ns	31.15 ns	20.73 ns

Table 2. Results for cache size of 256 bytes, block size of 1 byte and 16 KB main memory

However it can also be seen that there is not much difference between the three timing delay values obtained. This is due to the small size of the cache that we simulated. Simulation of our models using larger cache sizes (as seen in table 2) proved our belief that as the size of the cache increases, the improvement in performance of the difference-bit cache over the conventional set-associative cache increases proportionally. This is obvious, because as the cache size increases, the width of the tag array grows at a larger rate than that of the diff array which is $\log_2$ times the size of the tag width.

For the second simulation, we tried to synthesize a slightly larger cache with the following parameters:

Cache size: 256 bytes,
Block size: 1 byte

and we assumed a main memory of size 16 KB. The results of this synthesis are tabulated in table 2.

In our implementation we have considered a block size of one and a basic word size of only eight bits. But it can be easily seen that the size of the basic block would not alter the results we have obtained as this parameter would only increase the width of the tag array further resulting in the difference-bit cache faring even better against the conventional two-way set-associative cache. The basic word width does not affect the result either. An increase in this parameter would result in an increase in the time to access the selected word. However this increase would be seen to an equal extent in all the three cache models we are considering here.

Having access to only a slimmed down academic version of the Synopsys suite of tools, the overall access times that we obtained were higher than realistic values expected with state-of-the-art VLSI fabrication technologies. The use of better libraries in Synopsys, with lower component delays corresponding to state-of-the-art memory system VLSI technologies, would result in more realistic timing values. However our analysis is valid for comparison of the different cache schemes, as using better libraries would result in a somewhat equal downward scaling of the access times for all the three cache configurations.

3.3 Increase in area

The introduction of the *diff-memory* and associated circuitry increases the area requirements of the difference-bit cache when compared to the conventional set-associative cache. The main contribution to this area is the *diff memory* of size $NS \times b$ bits, whereas the data cache has $2NS \times L$ bits (L is the line size in bits) and the tag memory has an area $NS \times 2t$ so that the fraction of increase is

$$b/2(L + t)$$

Although we have not calculated the area contributions of all these sub-components of the cache in our Synopsys models, analysis has been presented [1] that puts the figure of fractional area increase at about 0.01 to 0.06 for a typical processor family like the Alpha.

3.4 Effect of Increased Associativity

The evaluation of the difference-bit cache presented so far has been restricted to an associativity of degree two. We have attempted to evaluate the effect of extending the implementation of this cache to attain a higher functional associativity. In the case of a four-way set-associative cache, each tag will have defined for it two diff-index positions (diff-index-0 and diff-index-1) which will be the bit positions in which that tag will vary from all the other three tags. The diff-values corresponding to these diff-indices will be also stored for each tag. Thus the width of an entry in the diff-array will be the sum of the width of eight diff-indices and the width of eight diff-values. This means a diff-array width of $\{8\log_2 t + 8\}$ where t is the size of the tag bits. This width will still be smaller than the width of the tag array ($4t$) as long as $t \geq 9$.

The enable logic will be a considerable variation from the two-way difference-bit case. When an address is forwarded by the processor, the select logic will check in the diff-index positions of tag 1 and compare the values in these positions with the values of the diff-values for tag 1. If there is a match, the address is present in the data block corresponding to tag 1. The select logic will perform this operation for each of the other tags (tag 2, tag 3 and tag 4) in parallel (as these operations are independent of each other). Although these comparisons can proceed in parallel, and even if the size of the diff-array is indeed considerably smaller than the tag-array, the complexity of the enable logic will result in a very small performance improvement, if any, over the conventional four-way set-associative cache.

The increase in memory area for achieving higher associativity will be at least about 0.05 to 0.20 in case of a typical processor like the Alpha [1]. The enable logic, although similar in nature to the degree two case, will now have to pass through a more complex decoder owing to the larger size of the *diff-array*. This will increase the access time of the cache by a factor of as much as 48 % (based on analysis of the cache access-time model by S. Wilton and N. Jouppi [2]). Also now, the logic required to compute this information in case of a cache miss will increase greatly. Considering the small improvement in miss ratio of a four-way set-associative cache over a two-way set-associative cache, increasing the associativity of the difference-bit cache beyond two, does not seem a very viable solution. A much better overall performance can be achieved by using a degree two difference-bit cache in the first level with a fully associative victim cache[10] or annex cache[11] as a cache assist.

4 Conclusions

In this paper we have modeled and analyzed a recent novel cache proposal called the difference-bit cache [1] and shown that it can achieve access times equal or close to those of a direct-mapped cache with the functionality of a two-way set-associative cache. This faster access time is obtained by separating the selection of the proper way from the detection of a hit and by selecting the way using the least-significant bit in which the two tags of a set differ. The performance of the difference-bit cache is better than the direct-mapped cache, the two-way set-associative cache and all other previous schemes that attempt to achieve direct-mapped access times in set-associative caches using prediction techniques.

We have also evaluated the effect of extending the difference-bit cache to higher degrees of associativity. The small decrease in miss ratio of higher associativity cache schemes over the two-way set-associative cache and the significantly larger increase in logic complexity and chip area for the associative difference-bit cache does not make this a viable proposition.

The increase in area for a degree two difference-bit cache is a very small fraction of the total chip area for all typical processor families. All these factors make the difference-bit cache, with its low access time and low miss ratio, an ideal candidate for the first level cache in today's high performance processors.

Acknowledgments

We thank Dr. Tomas Lang for his help in clarifying our understanding of the difference-bit cache and also for giving us helpful insights into the modeling of cache systems.

References

[1] Toni Juan, Tomas Lang and Juan J.Navarro, "The Difference-bit Cache", *International Symposium on Computer Architecture,* May 1996.

[2] Steven J.E.Wilton and Norman P.Jouppi, "An enhanced access and cycle time model for on- chip caches", *Research Report 93/5, Digital WRL,* Jul 1994.

[3] Lishing Liu, "Partial address directory for cache access", *IEEE Transactions on Very Large Scale Integration Systems,* Vol.2 (2): 226-239, Jun 1994.

[4] Kimming So and Rudolph N.Rechtschaffen, "Cache operations by MRU change", *IEEE Transactions on Computers,* Vol.37 (6): 700-709, Jun 1988.

[5] Andre Seznec, "DASC Cache", *Proceedings. of the 1st Intl. Symp on High-Performance Computer Architectures,* pg. 134-143, Jan 1995

[6] Gurinder S.Sohi, Scott E.Breach and T.N.Vijaykumar, "Multiscalar Processors", *International Symposium on Computer Architecture,* 1995.

[7] Jeffrey D.Gee, Mark D.Hill, Dionisios N.Pneumatikatos and Alan Jay Smith, "Cache Performance of the SPEC '92 benchmark suite", *IEEE Micro,* Vol. 13(4):8-16, Aug 1993.

[8] Mark D.Hill and Alan Jay Smith, "Evaluating Associativity in CPU Caches", *IEEE Transactions on Computers,* Vol. 38(12): 1612-1630, Dec 1989.

[9] C.Eric Wu, Yarsun Hsu and Yew-Huey Liu, "A quantitative evaluation of cache types for high-performance computer systems", *IEEE Transactions on Computers,* Vol. 42(10): 1154-1162, Oct 1993.

[10] Norman Jouppi, "Improving Direct-mapped cache performance by the addition of a Small Fully Associative Cache and Buffers", *Proceedings of the 17th Intl. Symposium on Computer Architecture,* pg. 364-373, 1990.

[11] L. John, A. Subramanian, "Design and Performance of a Cache Assist to implement selective caching", *Proceedings of ICCD,* 1997.