Way-lookup buffer for low-power set-associative cache

Sungjae Lee, Jinku Kang, and Inhwan Lee

School of Electrical and Computer Engineering, Hanyang University, Hangdang-Dong 17, Sungdong-Gu, Seoul, 133–791, Korea

Abstract: The way-predicting (WP) cache is simple and energy-efficient when used as L1 cache. However, a concern in the WP cache is the speed of the slow hit. This paper proposes the use of way-lookup buffer (WLB) cache to alleviate the problem of slow hit in the WP cache. The results show that the WLB slow hit is 33% faster than the WP slow hit in L1 cache. This is important because the slow hit is often the critical path that determines the processor clock cycle time. The WLB cache is as energy-efficient as the WP cache when used as L1 cache; using the WLB cache can reduce the power consumed by the conventional L1 cache by 68%. Therefore, if we like the nice properties (i.e., simplicity and energy efficiency) of the WP cache but are reluctant to use it because the slow hit extends the processor clock cycle time, then the WLB cache is the one to use.

Keywords: set-associative cache, low power, way-lookup buffer

Classification: Storage technology

References

1 Introduction

The way-predicting set-associative cache (WP cache) is simple and energy-efficient [1, 2]. The WP cache stores the way information of the most-recently-referenced (MRR) block for each set in a cache. Given a CPU request, it predicts a single way in which the requested data may exist by assuming that an MRR block is accessed again. If the prediction is correct, we can access only a single way and thus reduce the power consumption; otherwise, after accessing a single way based on the incorrect prediction, we have to access all other ways again. The former is called the fast hit and the latter is called the slow hit in this paper. The way prediction has been considered to be the most energy-efficient method for L1 cache. However, the slow hit is an important concern because it can directly affect the processor clock cycle time.

The phased cache divides the cache access in two phases [3]. In the first phase, it compares all tag arrays to determine whether it is a cache hit and also to find the way in which the requested data exists. If a cache hit occurs, only the selected way is accessed in the second phase. The WDU (way determination unit) cache maintains the tag and the way information of a few MRR blocks in a fully-associative cache called WDU [4]. If the information on the requested block exists in WDU, the WDU cache accesses only a single way indicated by the WDU. That is the WDU fast hit. Otherwise, it accesses all of the ways in a cache, which results in a slow hit.

In this paper, we propose the use of way-lookup buffer (WLB) in set-associative caches. Such a cache is called a WLB cache in this paper. The purpose of the WLB cache is to alleviate the problem of slow hit in the WP cache. This is important because the slow hit in L1 cache is often the critical path that determines the processor clock cycle time. This paper discusses the structure and operation of the WLB cache and then evaluates its performance.

2 Way-lookup buffer

Fig. 1 shows a 4-way set-associative cache with WLB. A WLB has as many entries as the number of sets in a cache. Each entry stores the tag as well as the way information of the MRR block for the corresponding set (or index). Given a CPU request, we use the index to access the WLB and extract the tag and way information. If the extracted tag and the tag of the requested data are identical, we have a WLB hit; otherwise, it is a WLB miss.

A WLB hit means that we are accessing the MRR block for the given index again. Obviously, a WLB hit also means a cache hit, and the information in WLB accurately identifies the way in which the requested data exists. Thus, we can access only a single way, which significantly reduces the power consumption. That is the WLB fast hit. (Note that, given a WLB hit, an additional tag comparison is not necessary.)

A WLB miss means that the requested block is not an MRR block. Then we skip accessing the way indicated by WLB and go directly to access the
remaining \((n - 1)\) ways in an \(n\)-way set-associative cache. If a cache hit occurs in the remaining \((n - 1)\) ways, that is a WLB slow hit. Then WLB is updated accordingly.

The purpose of the WLB cache is to alleviate the problem of slow hit in the WP cache. In a sense, the WLB cache is an extension of the WP cache. The only difference is that the WLB cache stores the tag as well as the way information of the MRR block while the WP cache maintains only the way information. Such difference does not have a significant impact on fast hit. However, the additional tag information in WLB makes an important difference in slow hit. Specifically, the tag information in WLB allows the WLB cache to access cache data blocks only once on a slow hit as discussed above. In contrast, the WP cache accesses cache data blocks twice; it first accesses a single way based on an incorrect way-prediction and then accesses \((n - 1)\) ways. The end effect is that the WLB slow hit can be significantly faster than the WP slow hit.

### 3 Experimental results

For performance evaluation, we first modify the SimpleScalar simulator [5] to implement the models of the WLB, WP, WDU, and phased caches. We use the Alpha 21264 processor model which is a 4-issue out-of-order superscalar processor. The processor has separate L1 instruction and data caches; each is a 4-way 64 KB cache with a block size of 64 B. It also has an 8-way 2 MB
L2 cache with a block size of 64 B. We assume the use of 180 nm technology and use CACTI 3.2 [6] to estimate the energy consumption, access time and hardware overhead. Finally, we run SPEC 2K integer applications with the Reference input set.

Let’s compare the WLB and WP caches in terms of L1 cache access latency (Table I). The numbers are the average values obtained by running the SPEC 2K integer applications. The most notable result is that the WLB slow hit is 33% faster than the WP slow hit. Therefore, when compared with the WP cache, the WLB cache can potentially reduce the processor clock cycle time by 33% because the clock cycle time is often determined by the slow hit. That is the key benefit of using the WLB cache. While the WLB fast hit is slower than the WP fast hit because accessing WLB takes slightly longer than way prediction, it is usually hidden because the clock cycle time is usually determined by the slow hit.

Note that the WLB hit rate and the way-prediction hit rate are identical (Table I). This happens because both the WLB hit and the way-prediction hit occur when an MRR block is accessed again. The WLB hit rate (or way-prediction hit rate) determines energy efficiency; it is quite high in L1 cache but drops significantly in L2 cache. The hardware overhead of the WLB cache is slightly higher than that of the WP cache, but the overhead is practically acceptable.

The phased cache is always slower than the WLB cache because of its two-phase access. The phased cache does not incur hardware overhead because it just modifies the cache access sequence. WDU and WLB contain the same information. However, WDU uses fully-associative mapping which is complex. To compare the WLB and WDU caches under the same hardware overhead, the WDU for L1 cache has 16 entries and the WDU for L2 cache has 256 entries in our simulation. Table I shows that the WDU cache is slower than the WLB cache, which is due to using fully-associative mapping. Also, since WDU contains fewer entries than WLB, the WDU hit rate is lower.

Table II shows the energy consumption and the energy-delay product
Table I. Relative energy consumption and energy-delay product (%).

<table>
<thead>
<tr>
<th></th>
<th>L1 Instruction</th>
<th>L1 Data</th>
<th>L2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conventional cache</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>WLB cache</td>
<td>31.42</td>
<td>weighted: 35.7 fast: 35.0 slow: 104.9</td>
<td>33.38</td>
</tr>
<tr>
<td>WP cache</td>
<td>31.69</td>
<td>weighted: 33.7 fast: 32.4 slow: 155.9</td>
<td>34.67</td>
</tr>
<tr>
<td>Phased cache</td>
<td>49.50</td>
<td>94.1</td>
<td>49.04</td>
</tr>
<tr>
<td>WDU cache</td>
<td>34.40</td>
<td>weighted: 59.7 fast: 51.6 slow: 191.5</td>
<td>43.81</td>
</tr>
</tbody>
</table>

normalized by those of the conventional cache. Let’s consider the L1 cache. The table shows that the WLB cache is as energy-efficient as the WP cache. Using the WLB cache can reduce the power consumed by the conventional L1 cache by 68%. Such high energy efficiency is attributed to the strong locality in L1 cache. If we examine more closely, the advantage of the WLB cache is that it accesses only \((n - 1)\) ways in \(n\)-way set-associative caches on a slow hit. In contrast, the WP cache accesses all \(n\) ways. But then the WLB cache stores the tag information of all MRR blocks as well. When all factors are combined, the WLB cache turns out to be slightly more energy-efficient than the WP cache.

Table II shows the energy-delay products which are calculated based on “fast” hit, “slow” hit, and the “weighted” average of the fast and slow hits. The energy-delay product of the WLB cache for fast hit is slightly higher than that of the WP cache. However, given that the WLB and WP caches consume a similar amount of energy especially on a fast hit, this result simply reflects the fact that the WP fast hit is faster than the WLB fast hit. On the other hand, the energy-delay product of the WLB cache for slow hit is significantly better than that of the WP cache. This is because the WLB cache is faster and more energy-efficient on a slow hit.

The phased and WDU caches are not energy-efficient when used as L1 cache (Table II). As for L2 cache, the phased cache is the most energy-efficient because slow hits are common in L2 cache. However, the phased cache has long access latency (Table I). So there is a trade-off in L2 cache. We can use the WLB cache if both short access latency and energy efficiency are important; we can use the phased cache if energy efficiency is the utmost concern. The WDU cache is neither fast nor energy-efficient.

4 Conclusion

The WP cache is simple and energy-efficient when used as L1 cache. However, a concern in the WP cache is the speed of the slow hit. In this paper, we present the WLB cache to alleviate the problem of slow hit in the WP cache.
The results show that the WLB slow hit is 33% faster than the WP slow hit in L1 cache. This is important because the slow hit is often the critical path that determines the processor clock cycle time. The WLB cache is as energy-efficient as the WP cache when used as L1 cache; using the WLB cache can reduce the power consumed by the conventional L1 cache by 68%. Therefore, if we like the nice properties (i.e., simplicity and energy efficiency) of the WP cache but are reluctant to use it because the slow hit extends the processor clock cycle time, then the WLB cache is the one to use. That is the key contribution of the paper. In the case of L2 cache, we can use the WLB cache if both short access latency and energy efficiency are important; we can use the phased cache if energy efficiency is the utmost concern.