Concurrent Support of Multiple Page Sizes On a Skewed Associative TLB

André Seznec

to appear in IEEE Transactions on Computers

9 october 2003

Author contact information

André Seznec
tel: +33 2 99847336
fax: +33 2 99847100
e-mail: seznec@irisa.fr
address: IRISA/INRIA, Campus de Beaulieu, 35042 Rennes, France

Abstract

Some architecture definitions (e.g. Alpha) allows the use of multiple virtual page sizes even for a single process. Unfortunately, on current set-associative TLBs (Translation Lookaside Buffers), pages with different sizes can not coexist together. Thus, processors supporting multiple page sizes implement fully-associative TLBs.

In this research note, we show how the skewed-associative TLB can accommodate the concurrent use of multiple page sizes within a single process. This allows to envision either medium size L1 TLBs or very large L2 TLBs supporting multiple page sizes.

Keywords

TLB, multiple page size, skewed associativity.

1 Introduction

The performance of an application is often very dependent on the virtual memory behavior. Optimal performance of different applications requires the use of different page sizes. Different page size granularities (for different code/data regions) are also often beneficial for a single process. Using small page size avoids to allocate physical memory space for unused data or instructions. On the other hand, using a large page size results in a single page fault for a large region of data or instructions. Virtual-to-physical address translation is handled by the Translation Lookaside Buffer (TLB). With many applications TLB miss handling constitutes a very significant portion of the execution time [1, 2, 3, 4]. At equal number of TLB entries, a large page size leads to a smaller number of TLB misses than the one induced by a small page size. In order to deal with the page size issue, some architecture definitions (e.g. Compaq Alpha [8]) provides support for multiple page sizes. A few techniques have been proposed to use different page sizes in a single process [11, 10, 9].

Unfortunately, set-associative TLBs are not efficient to support concurrent multiple page sizes for a single process. Therefore, whenever an architecture allows the concurrent use of multiple page sizes, the processor implements either fully associative TLB or distinct TLBs. This stands for level one TLBs as well as level two TLBs.

In this research note, we present a skewed associative TLB supporting concurrent multiple page sizes. The structure is derived from the skewed associative cache [5, 7]. With an architecture supporting X page sizes, the proposed TLB requires at least 2^{n+1} ways with 2^n being the first power of two larger than X. For example, considering the example of the Alpha architecture that supports 8 Kbytes, 64 Kbytes, 512 Kbytes and 4 Mbytes pages, our structure requires a 8-way 1024-entry TLB for supporting the four page sizes. Using such a 8-way TLB has to be contrasted against either using a fully-associative TLB or using distinct TLBs for supporting the four page sizes. If a single page size is used then the behavior of the TLB is the one of a 2-way skewed associative 1024-entry one-page size TLB, and therefore, is close to the behavior found with a 8-way set-associative 1024-entry TLB [6].

Our solution has clear advantages over both of the previous proposals to concurrently support multiple page sizes.

- **Use of fully associative TLBs:** use of fully associative structures is limited to a small number of entries (64-128). Using a skewed associative TLB allows to build a large TLB supporting concurrent multiple page sizes.

- **Use of distinct partially associative TLBs for different page sizes**: This solution is sub-optimal when the distribution of page sizes is not what the designer predicted. For instance, if four page sizes are supported and four equal sizes TLBs are used then an application using a single virtual page will benefit from only one fourth of the total number of entries in the TLB. Our solution allows this application to use all the entries in the TLBs.

The remainder of this research note is organized as follows. Section 2 shows how a fully-associative TLB can support multiple page sizes and why set-associative TLBs can not support them. Section 3 presents our proposal for supporting multiple page size with skewed associative TLB. Section 4 summarizes the study.

2 Concurrent use of multiple page sizes and TLBs

A few architecture definitions allow the concurrent use of several page sizes by a single process. For instance, the Alpha architecture defines four page sizes. The IA32 definition allows the use of 4 Mbytes pages in concurrency with the usual 4K page size. With such an architecture, the virtual-to-physical address translation of the address of word A computes two unknown data: the page size and the physical address of the page.

Ideally, the virtual-to-physical address translation is resolved through the TLB. When a single page size is used, there is no ambiguity in defining the page number for a virtual address A. The page number is presented to the TLB and it is checked against all possibly corresponding entries in the TLB, (i.e, all entries in a fully associative TLB or an entry in each way on a X-way set-associative TLB).

This process can be adapted quite simply to support multiple page sizes on a fully-associative TLB. Let $TAG_{tobechecked}$ be the page tag number if the effective page size was the minimum page size allowed by the architecture. $TAG_{tobechecked}$ is presented for comparison to any entry in the TLB. Each entry in the TLB features the tag number TAG_{mapped} of the mapped page (i.e the page number completed with zeros in order to match the number of bits required by the minimum page size). The TLB entry also features a mask (or an encoded mask) $Mask_{mapped}$ to assess the size of the mapped page. Tag check is performed as follows. The page tag number $TAG_{tobechecked}$ is masked with $Mask_{mapped}$ and, then compared with TAG_{mapped}.

The difficulty with the set-associative TLB is that the page size and therefore the virtual page number for a virtual address is not known before address translation. With a set-associative TLB, a page can only be mapped in a single set. Therefore, the index of this set can only use the address bits that are common to all possible page numbers: for instance, if two page sizes are allowed 8 Kbytes and 4 Mbytes, the index in the TLB must ignore the 22 least significant bits of the address, and 512 contiguous small pages will conflict on a single set of the TLBs. Such a solution is clearly unacceptable for performance reasons.

3 Supporting multiple page sizes with a skewed associative TLB

In this section, we first recall the structure of the skewed associative cache. Then we present how to support multiple page sizes with a skewed associative TLB. We illustrate our proposal with the Alpha architecture that supports 4 different page sizes.

3.1 Skewed-associative cache principles

Skewed associative caches were proposed in [5, 7] in order to reduce conflict misses with partially associative caches. A X-way set-associative cache is built with X distinct banks as illustrated in Figure 1 . The memory block at address D may be physically mapped onto physical line $f(D)$ of any of the distinct banks. This vision of a set-associative cache fits with the physical implementation of X banks of static RAMs.

For a skewed associative cache (Figure 2), a different mapping function is used for each cache bank: A memory block at address D may be mapped onto physical line $f_0(D)$ in bank 0, onto physical line $f_1(D)$ in bank 1, etc.

It has been shown in [5, 7] that, provided that the indexing functions exhibit some distribution properties, skewed-associative caches exhibit an average lower miss ratio than set-associative caches with general purpose applications.

3.2 A skewed associative TLB supporting multiple page sizes

3.2.1 Principle

We adapt the skewed associative structure to TLBs in order to support concurrent multiple page sizes.

A skewed associative TLB concurrently supporting multiple page size has the three following properties:

Property 1 Each virtual address has a possible and unique mapping location in each way. More precisely, for any given way I in the TLB, for any given virtual address V(A), there exists a single entry E in way I and there exists at least one possible page size s, such that the virtual page number associated with V(A) and s can be mapped on entry E in way I.

Property 2 The distinct ways of the cache are indexed with different hashing functions F_i.

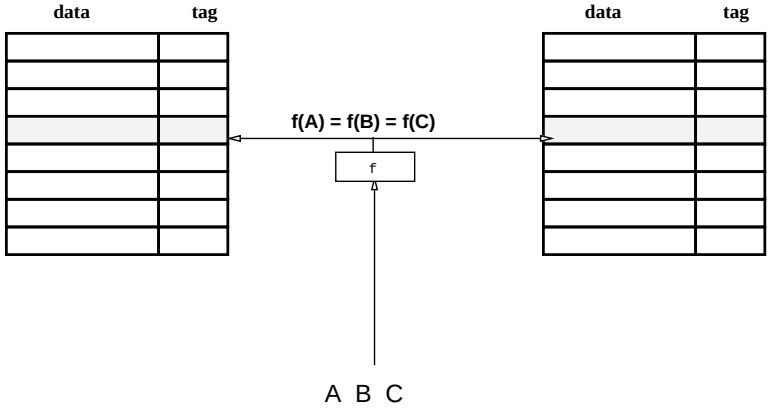

Figure 1: 3 data blocks conflicting for a single set on a two-way set-associative cache. A, B and C compete for only two locations

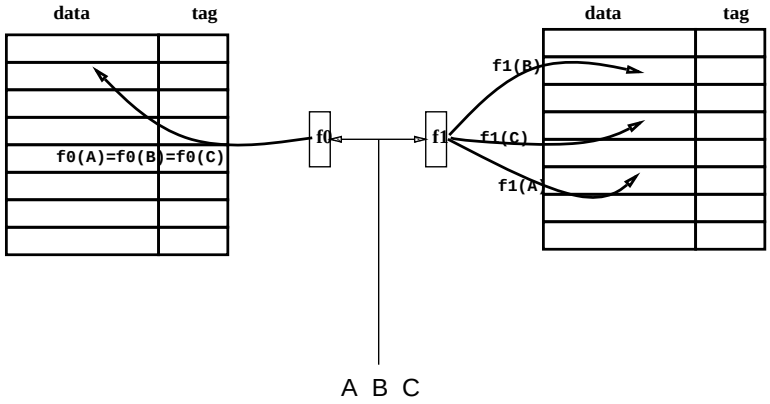

Figure 2: A, B and C compete for the same location in bank 0, but can be present at the same time, as they do not map to the same location in bank 1

Property 3 For any of the supported page sizes s, applications using s as its only page size can use the over all space of the TLB.

Property 1 is associated with the definition of partially associative caches. Property 2 is the definition of skewed associativity. Property 3 can be seen as the definition of concurrently supporting multiple page size. It further means that any TLB location can map pages of any possible size.

The difficulty with property 1 is that the virtual page number associated with a virtual address is unknown: it depends on the page size. To turn this difficulty, we enforce the following constraint:

> There exists a page size function S such that, for any given way I of the TLB and for any given word at virtual address A, virtual page $V(A)$ of word A can be mapped on way I if and only if the size of the virtual page is $S(A, I)$.

This constraint means that, if the virtual page $V(A)$ is mapped on a given way I from the TLB then its page size is known and therefore one can define $P(A/I)$, the page number of virtual page $V(A)$ knowing that virtual page $V(A)$ is mapped by way I in the TLB. The unique location on way I where page $V(A)$ can then be computed from this virtual page number.

An example of a skewed associative TLB respecting the 3 properties (i.e. concurrently supporting multiple page sizes) is illustrated below for the Alpha example. For other examples, one can use the general approach developed in this example.

3.2.2 An example for the Alpha architecture

The Alpha ISA [8] supports four page sizes: 8 Kbytes, 64 Kbytes, 512 Kbytes and 4M pages. We illustrate a possible design for a 8-way 1024-entry skewed associative TLB supporting these four page sizes.

We first propose a possible page size function and then propose a set of associated index functions for the ways of the TLB.

Page size function A possible page size function S is illustrated on Figure 3. S only depends on the chain X(A) of the 3 bits 21 to 23 from the address A (bit 0 being the least significant bit). S is formally defined below:

- **ways 0 and 4**: 8K for X(A)=0,4; 64K for X(A)= 1,5; 512K for X(A)= 2,3; 4M for X(A) = 6,7.
- **ways 1 and 5**: 8K for X(A)=1,5; 64K for X(A)= 0,4; 512K for X(A)= 6,7; 4M for X(A) = 2,3.
- **ways 2 and 6**: 8K for X(A)=2,6; 64K for X(A)= 3,7; 512K for X(A)= 0,1; 4M for X(A) = 4,5.
- **ways 3 and 7**: 8K for X(A)=3,7; 64K for X(A)= 2,6; 512K for X(A)= 4,5; 4M for X(A) = 0,1.

Let us analyze the properties of the proposed page function S:

1. **Page size distribution on a single way**. A given way I must map the entire virtual address space (Property 1). Any TLB location must also be possibly mapped by a page of any of the four different sizes (Property 3).

   To respect these properties, we have forced each given way I to map one fourth of the virtual address space with page size 8 Kbytes, a second fourth with page size 64 Kbytes, a third fourth with page size 512 Kbytes and the last fourth with page size 4 Mbytes.

2. **Virtual space distribution over the ways for a fixed page size**. For a given page size s, each way maps only one fourth of the total virtual space. The total virtual space must be mapped by the TLB for each of the possible page sizes. In order to enforce this property, we have imposed that any page V(A) with size s has one and only one TLB location target in one of the ways numbered from 0 to 3 (and a second location target in one and only one of the ways numbered from 4 to 7).

An example of indexing functions The 8 indexing functions f_i for the 8 ways of the skewed-associative TLB can be different. We build such an example below.

For a virtual address A and any of the ways I in 0 to 3, the index consists of the 7 least significant bits of the page number $P(A/I)$ excluding the two "way selection" bits, i.e.,bits 21 and 22 for 8 Kbytes and 64bytes page sizes, and bits 22 and 23 for 512 Kbytes and 4 Mbytes page sizes. For ways 4 to 7, we exclusive-OR these least significant bits with 7 extra bits in the page number.

A more formal definition of these skewing functions f_i is given below:

let N be the number of bits in the virtual address,
let $(A_{N-1}, .., A_0)$ be the binary representation of a virtual address A,
let us consider the function $c(k, A)$ defined by $c(k, A) = (A \ mod \ 2^k) + ((A/2^{k+2}) * 2^k)$ (c removes bits k and k+1 from A),
let us consider the function h(k,A) defined by $h(k, A) = (c(k, A)/2^k) \ mod \ 128$,
let us consider the function H(k,A) defined by $H(k, A) = h(k, A) \oplus h(k + 7, A)$ where $\oplus$ is the exclusive OR
let us consider the function g on the set $\{0, .., 2^{N-1} - 1\} * \{8K, 64K, 512K, 4M\}$, defined by g(A,8K) = h(13,c(21,A)), g(A,64K)= h(16,c(21,A)), g(A,512K)= h(19,c(22,A)) and g(A,4M)= h(22,c(22,A))
let us consider the function G on the set $\{0, .., 2^{N-1} - 1\} * \{8K, 64K, 512K, 4M\}$ defined by G(A,8K) = H(13,c(21,A)), G(A,64K)= H(16,c(21,A)), G(A,512K)= H(19,c(22,A)) and G(A,4M)= H(22,c(22,A))
the 8 indexing functions f_i for the 8 ways of the skewed-associative TLB are defined by :

- for i, $0 \leq i < 4$, $f_i(A) = g(A, S(A, i))$
- for i, $4 \leq i < 8$, $f_i(A) = G(A, S(A, i))$

Effective associativity For a given page size, any virtual page V(A) has two possible locations in the TLB. That is, the effective associativity degree of the TLB is limited to two, since V(A) has only one page size. Nevertheless it was shown in [6] that, in practice, the behavior of 2-way skewed-associative TLB is close to the one found with a 8-way set-associative TLB.

Overhead in index computation Computing the skewing functions $(f_i)_{i<4}$ is very simple. Four fixed 7-bit strings are extracted from the virtual address and are presented at the entry of a 4-to-4 multiplexor. Control of the multiplexor is given by the page size function S i.e, is derived from the 3 virtual address bits A_{21}, A_{22} and A_{23}.

An extra 2-entry exclusive-OR gate level is needed to compute the functions $(f_i)_{i\geq 4}$.

About replacement policy For a given virtual page with a fixed page size, there exists only two possible locations in the TLB. Any of the previously proposed replacement policies for 2-way skewed associative cache can be used [6].

About the number of entries in the TLB With current processor implementations, a N-entry TLB supporting a page size s is able to map a contiguous region of size $s * N$.

It is also desirable that a TLB supporting multiple page sizes can map a large contiguous region of the virtual memory. In the example of the Alpha architecture described here, we have forced this property by assuming 1024 entries, i.e 2 times the ratio of maximum page size on minimum page size.

To determine the ways where the virtual page of address A can be mapped with page size 4 Mbytes, one cannot use any of the 22 least significant bits. Therefore we use bits A_{22} and A_{23}. For a virtual page using

minimum page size 8 Kbytes, bit A_{22} is used to determine the parity of the numbers of the ways that can map the page.

And if there were only three possible page sizes? Let us consider the case where only three page sizes have to be supported. Let us say 8 Kbytes, 512 Kbytes and 4 Mbytes.

In this case, we still can use a 8-way skewed-associative TLBs. The smaller page granularity can be favored by allowing an effective associativity degree of four for this page granularity.

Constraints on the page size function are slightly changed:

- **Page size distribution on a single way**: one half of the virtual address space is mapped with page size 8 Kbytes, a fourth with page size 512 Kbytes and the last fourth with page size 4 Mbytes.
- **Virtual space distribution over the ways for a fixed page size**. a) any virtual page V(A) with size 8Kbytes can be mapped onto one of the two ways $2 * i$ to $2 * i + 1$ b) any virtual page V(A) with size 512 Kbytes or 4 Mbytes can be mapped onto one of four ways 0 to 3 (respectively 4 to 7).

In practice, our solution can be adapted to support concurrently $2^{m-1} < M \leq 2^m$ different page sizes while providing some associativity on 2^{m+1} way skewed associative TLB.

4 Conclusion

Concurrently supporting multiple page sizes with a single process allows a better management of the virtual memory than supporting a single page size. Current set-associative structures for TLBs are not able to support concurrently multiple page sizes for a single process: therefore, whenever an architecture allows the concurrent use of multiple page sizes, the processor implements either (relatively small) fully associative TLBs or a distinct TLBs supporting the different page sizes.

In this research note, we have proposed a skewed associative TLB that concurrently supports multiple page sizes: any location from the TLB can map pages with any of the possible page size. However the effective associativity of the TLB, i.e the number of possible locations for a given virtual page with a given page size is smaller than the number of ways in the TLB (2 against 8 in the example illustrating this paper).

Such skewed-associative TLBs could be used either to implement medium size partially associative level one TLBs or partially associative large size level two TLBs.

References

[1] Thomas E. Anderson, Henry M. Levy, Brian B. Bershad, and Edward D. Lazowska. The interaction of architecture and operating system design. In *Proceedings of the 4th International Conference on Architectural Support for Programming Languages and Operating Systems*, Santa Clara, CA, USA, April 1991.

[2] J. Huck and J. Hays. Architectural support for translation table management in large address space machines. In Lubomir Bic, editor, *Proceedings of the 20th Annual International Symposium on Computer Architecture*, pages 39–51, San Diego, CA, May 1993.

[3] Mendel Rosenblum, Edouard Bugnion, Scott Devine, and Stephen Alan Herrod. Using the SimOS machine simulator to study complex computer systems. *ACM Transactions on Modeling and Computer Simulation*, 7(1), January 1997.

[4] Ashley Saulsbury, Fredrik Dahlgren, and Per Stenström. Recency-based TLB preloading. In *Proceedings of the 27th Annual International Symposium on Computer Architecture*, Vancouver, British Columbia, June 12–14, 2000.

[5] André Seznec. A case for two-way skewed-associative caches. In *Proceedings of the 20th Annual International Symposium on Computer Architecture*, San Diego, CA, May 1993.

[6] André Seznec. A new case for skewed-associativity. Technical Report RR-3208, Inria, Institut National de Recherche en Informatique et en Automatique, July 1997.

[7] André Seznec and Francois Bodin. Skewed-associative caches. In *Proceedings of PARLE '93 – Parallel Architectures and Languages Europe*, Lecture Notes in Computer Science, Munich, Germany, June 14–17, 1993.

[8] Richard L. Sites. *Alpha Architecture Reference Manual*. Digital Press and Prentice-Hall, 1992.

[9] M. Swanson, L. Stroller, and J. B. Carter. Increasing TLB reach using superpages backed by shadow memory. In *Proc. of the 25th Annual Int'l Symp. on Computer Architecture (ISCA'98)*, June 1998.

[10] Madhusudhan Talluri and Mark D. Hill. Surpassing the TLB performance of superpages with less operating system support. In *Proceedings of the Sixth International Conference on Architectural Support for Programming Languages and Operating Systems*, San Jose, California, October 4–7, 1994.

[11] Madhusudhan Talluri, Shing Kong, Mark D. Hill, and David A. Patterson. Tradeoffs in supporting two page sizes. In *Proceedings the 19th Annual International Symposium on Computer Architecture,ACM SIGARCH*, pages 415–424, Gold Coast, Australia, May 1992.

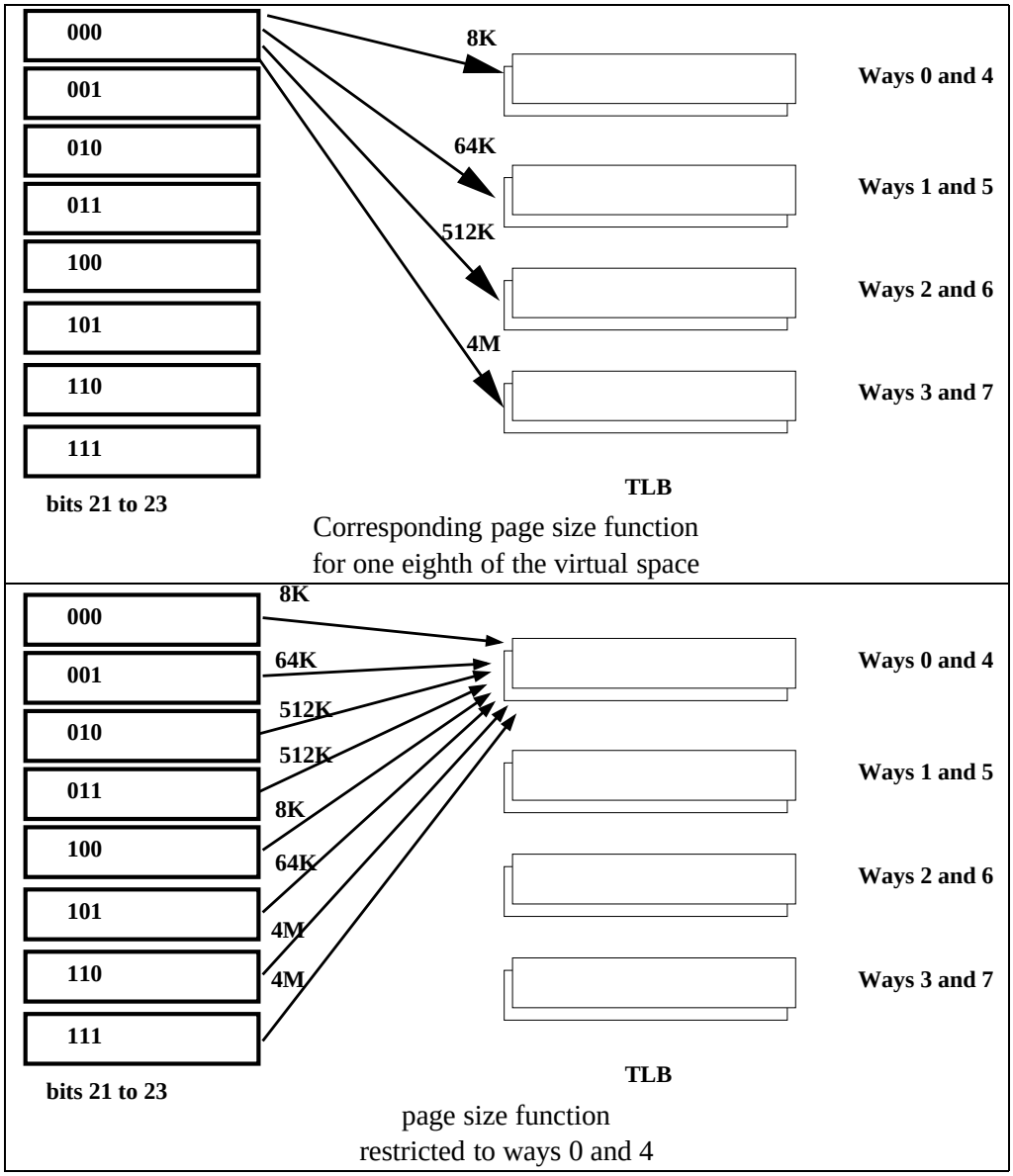

Figure 3: Mapping the virtual space on the TLB way