Don't Use the Page Number, but a Pointer on It

André Seznec

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Don't use the page number, but a pointer on it

André Seznec

N° 2727

Novembre 1995

PROGRAMME 1

ISSN 0249-6399

Don't use the page number, but a pointer on it

André Seznec *

Programme 1 — Architectures parallèles, bases de données, réseaux et systèmes distribués
Projet Caps

Rapport de recherche n ° 2727 — Novembre 1995 — 22 pages

Abstract: Most newly announced microprocessors manipulate 64-bit virtual addresses and the width of physical addresses is also growing. As a result, the relative size of the address tags in the L1 cache is increasing. This is particularly dramatic when small block sizes are used. At the same time, the performance of complex superscalar processors depends more and more on the accuracy of branch prediction, while the size of the Branch Target Buffer is also increasing linearly with the address width.

In this paper, we apply the very simple principle enounced in the title for limiting the tag size of on-chip caches, and for limiting the size of the Branch Target Buffer. In an indirect-tagged cache, the anachronic duplication of the page number in processors (in TLB and in cache tags) is removed. The tag check is then simplified and the tag cost does not depend on the address width. Then applying the same principle, we propose the Reduced Branch Target Buffer. The storage size in a Reduced Branch Target Buffer does not depend on the address width and is dramatically smaller than the size of the conventional implementation of a Branch Target Buffer.

Key-words: address width, tag implementation cost, indirect-tagged caches, Reduced Branch Target Buffers

(Résumé : tsvp)

This work was partially supported by CNRS (inter-PRC project ILIAD)

*seznec@irisa.fr

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex (France)
Téléphone : (33) 99 84 71 00 – Télécopie : (33) 99 84 71 71

N'utilisez pas le numéro de page, mais un pointeur sur lui

Résumé : La plupart des microprocesseurs annoncés récemment manipulent des adresses virtuelles 64-bit ; dans le même temps, la largeur des adresses physiques croît de la même manière. Il en résulte que la taille des étiquettes dans les caches de premier niveaux croît. Ceci est particulièrement dramatique quand des blocs de petites tailles sont utilisés. Dans le même temps, les performances des processeurs superscalaires dépendent de plus en plus de la qualité de la prédiction de branchement tandis que la taille du Tampon de Cibles de Branchement (BTB) croît de manière linéaire avec la largeur de l'adresse.

Dans cet article, nous appliquons le principe extrèmement simple énoncé dans le titre pour limiter la taille des étiquettes sur les caches *on-chip* et la taille du BTB.

Sur un cache indirectement étiqueté, la duplication anachronique du numéro de page dans les processeurs (dans le TLB et dans les étiquettes du cache) est supprimée. Appliquant le même principe, nous proposons ensuite le *Reduced Branch Target Buffer*. Le volume de mémorisation dans le Reduced Branch Target Buffer ne dépend pas de la largeur de l'adresse et est très inférieur à la taille d'un BTB implanté de manière classique.

1 Introduction

Information stored in caches consists of data and tags. Address tags allow to retrieve the address of a block, while validity, dirty and coherency tags allow to maintain data consistency in the distinct levels of a memory hierarchy system. Most announced microprocessors manipulate 64-bit virtual addresses while the width of physical addresses is also growing. As a result, the size of the address tags in traditional caches is increasing. This is particularly dramatic when small data block sizes are used. Then tag implementation cost becomes an important issue for on-chip caches.

Performance of superscalar microprocessors depends on the accurary of dynamic branch prediction. Large Branch Target Buffers must be used in processors in order to allow very accurate prediction. For a Branch Target Buffer, the implementation costs for both tags (i.e. branch addresses) and data (i.e. target addresses) increase linearly with the address width. The silicon area occupied by a Branch Target Buffer can become quite significant. Decreasing this area is also becoming an issue for microprocessor designers.

Paradoxaly, in current microprocessors, the page number associated with a cache block is represented two or three times, first as a part of the address tag associated with the cache block, second as a page address in a TLB entry for virtual-to-physical address translation, and sometimes a third time as a parcel of the branch address or the target address. It is even more curious that these informations are read at the same time *and* compared during the tag check.

The contribution of this paper is to show that these anachronistic replications of information may be removed by applying the simple following principle:

Do not use the page number, but a pointer on it

The remainder of the paper is organized as follows. In Section 2, we point out how dramatic can be the hardware cost of L1 cache address tags and of Branch Target Buffers. In Section 3, we propose an elegant solution for limiting the tag size for on-chip caches, *the indirect-tagged cache*. In the indirect-tagged cache, the page offset in an address tag is replaced by a pointer to a page table. This page table may be the TLB for virtually tagged caches or a physical page table. The tag check using an indirect-tagged physically tagged cache is simpler than with conventional cache designs. The size of the tag array in an indirect-tagged cache does not depend on the address width. The cache miss ratios of indirect-tagged caches are shown to be very close to the cache miss ratios of conventional caches. The execution overhead associated with the use of an indirect tagged cache is also shown to be small. In Section 4, we present a similar solution for the Branch Target Buffer. The size of our *Reduced Branch Target Buffer* does not depend on the address width. It is shown that with current parameters this size is 60 % lower than for a conventional BTB while providing an equivalent target prediction ratio. Section 5 concludes this study.

Related work

Reducing tag implementation costs In order to conciliate small or medium line size with a low tag array size, many cache designers have used sectored caches [6]. A cache sector consists of several contiguous cache lines; each cache line has its own coherency and validity tags, but all the cache lines in a cache sector share a single address tag. When two cache blocks are valid in a single cache sector, then they belong to the same memory sector (i.e. their addresses only differ by the sector offset). The tag array size on a sectored cache with sector size L is approximately the same

as the tag array size on a traditional cache with block size L. But on a sectored cache, the transfer granularity from memory to the processor is a cache block. Goodman [4] pointed out that the low granularity of data transfer associated with sectored caches reduces bus traffic. In a coherent shared memory multiprocessor, the bus may become the performance bottleneck as it is used for maintaining coherency of caches. The probability of ping-pong phenomena due to false sharing is lower with small cache blocks (an advantage of using cache sectors) than with larger blocks [2]. However for many applications, a sectored cache yields in a higher miss ratio than a non-sectored cache [4, 16].

In [16], Seznec presented the decoupled sectored caches. The principle is to dissociate the address tag array from the data array. A pointer named selection tag is associated with each cache block, this selection tag points on an address tag location and allow to retrieve the memory address of the block. The presentation of decoupled sectored caches was focussed on better cache usage with equivalent tag implementation cost. In [16], the behavior of external L2 caches was mainly addressed. In this paper, we address on-chip cache and Branch Target Buffer issues.

In [19], Wang et al. proposed a technique for reducing area cost of on-chip caches (Caching Address Tags). The presentation of Caching Address Tags focussed on reducing the tag implementation cost by exploiting data locality. Wang et al. remarked that the address tags stored in a cache are often equal. They suggested not to associate the address tag to the data block, but to use a pointer to a specialized table named Caching Address Tags where the effective address tag is stored. Caching Address Tags [19] may be interpreted as a decoupled sectored cache where the sector size is equal to the size of an associativity degree of the cache.

In [18], Suzuki et al. proposed the TLB-unified cache (TUC). In a TUC, a pointer to a TLB entry is stored in the address tag array instead of the address tag. The TLB-unified cache may be a solution for virtually tagged caches. The indirect-tagged cache presented in this paper addresses also physically tagged caches.

Reducing branch target buffer implementation cost Calder and Grunwald [1] proposed the NLS architecture (next cache line and set architecture). Their study completed and formalized a previous study by Johnson [9].

Usual branch prediction mechanisms deliver a complete address. But, what is really needed is the position in the cache of the next instruction to be fetched. The NLS principle consists in only storing this useful information in a tag-less direct-mapped table instead of the complete address thus saving storage area. The simulation results presented in [1] showed that, at comparable implementation cost, NLS tables have better target prediction ratios than conventional branch target buffers (BTBs).

2 The address width problem

2.1 Tag implementation cost in conventional caches

In conventional caches, a tag word is associated with each cache line. This tag word consists of an address tag and some other status tags (figure 1); the address tag allows to retrieve the effective address (virtual or physical) of the data block stored in the cache line. Due to 64-bit architectures virtual and physical addresses are now very wide. For instance, the address tag on the 8Kbytes physically indexed L1 direct-mapped cache of the DEC 21164 is 27-bit wide.

The address tag represents the major part of the tag word. The width of the tag word becomes quite significant compared to the width of a cache block when the block size is small (e.g. 16 bytes).