# Direct-mapped versus : set-associative pipelined caches

Nathalie Drach, André Seznec, Daniel Windheiser

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET AUTOMATIQUE

# *Direct-Mapped Versus Set-Associative Pipelined Caches*

Nathalie Drach, André Seznec, Daniel Windheiser

**N° 2256**

Mars 1994

PROGRAMME 1

Architectures parallèles,

bases de données,

réseaux et systèmes distribués


ISSN 0249-6399

# Direct-Mapped Versus Set-Associative Pipelined Caches

Nathalie Drach, André Seznec, Daniel Windheiser *

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

Rapport de recherche n ° 2256 — Mars 1994 — 19 pages

**Abstract:** As the tag check may be executed in a specific pipeline stage, cache pipelining allows to reach the same processor cycle time with a set-associative cache or a direct-mapped cache. On a direct-mapped cache, the data or the instruction flowing out from the cache may be used in parallel with the tag check. When using a pipelined cache, such an optimistic execution results in load and branch delays one cycle shorter than on a associatifs parv ensemble cache with the same pipeline depth.

In this paper, pipelined set-associative caches and pipelined direct-mapped caches using optimistic execution are compared. Our experiments show that for cache sizes in the 4K-16Kbytes range, the set-associative caches outperform the direct-mapped caches with current microprocessor miss penalty and a cache pipeline depth lower than 4 cycles. The gap between performance levels respectively obtained with set-associative caches and direct-mapped caches is particularly significant when a dynamic prediction branch strategy is used.

**Key-words:** pipelined caches, optimistic execution, code scheduling, branch delay, load delay

*(Résumé : tsvp)*

*Site Expérimental en Hyperparallélisme (SEH), Etablissement Technique Central de l'Armement (ETCA), 16 bis, Av. Prieur de la Côte d'Or, 94114 Arcueil Cedex

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

# Comparaison des caches pipelinés à correspondance directe et associatifs par ensemble

**Résumé :** La comparaison d'étiquette pouvant être exécutée dans un étage spécifique du pipeline, la mise en oeuvre de cache pipeliné permet d'obtenir un temps de cycle identique pour les caches à correspondance directe et les caches associatifs par ensemble. Pour les caches à correspondance directe, l'utilisation des données ou des instructions lues est effectuée en parallèle avec la comparaison d'étiquette. Cette utilisation anticipée permet de réduire d'un cycle le délai de branchement par rapport aux caches associatifs par ensemble.

Dans ce papier, nous comparons les performances des caches pipelinés associatifs par ensemble et des caches pipelinés à correspondance directe utilisant l'exécution optimiste presentée ci-dessus. Nos expériences ont montré que pour des tailles de cache variant de 4 Koctets à 16 Koctets, les caches associatifs par ensemble donnent de meilleures performances que les caches à correspondance directe pour des profondeurs de pipeline du cache inférieures à 4 cycles. Cette différence de performance est particulièrement significative lorsqu'une stratégie de prédiction de branchement est mise en oeuvre.

**Mots-clé :** caches pipelinés, exécution optimiste, ordonnancement de code, délai de branchement, délai de chargement

# 1 Introduction

Processor performance can be improved through cache pipelining to decrease the clock frequency and through increasing cache associativity to decrease the miss ratio. However pipeline organization and cache associativity are not independent, therefore these two aspects have to be addressed simultaneously when optimizing a pipeline.

As on-chip cache misses induce long pipeline stalls, reducing the miss rate has become a major issue for microprocessor designers [7]. The main argument for using a set-associative cache rather than a direct-mapped cache is a better hit ratio. In [3, 5], the authors reported that using a two-way set-associative cache instead of a direct-mapped cache removes about 30% of the misses. Generally the processor cycle time is determined by the cache hit time[1]. In most microprocessor designs, the cache hit time is one clock cycle. But generally, in these microprocessors, access to the cache is the longest pipeline stage, thus it determines the clock cycle, because all stages must proceed at the same rate.

But the cache hit time varies with the cache associativity. A direct-mapped cache access can be decomposed into two steps:

1. Read the word and the associated tag.
2. Check the tag against the data address.

A $n$-way set-associative cache access can be decomposed into three steps (see Figure 1):

1. Read a set of $n$ words and their associated tags.
2. Check the $n$ tags in parallel against the data address.
3. Select the correct word in the set.

In a direct-mapped cache, data (or instruction) flowing out of the cache may be directly used after step 1, while the tag check can be performed in parallel with other pipeline activities during the next cycle (see Figure 2). On a miss, this pipeline cycle must be canceled. We shall refer to this technique as *optimistic execution*. Using *optimistic execution*, a direct-mapped cache hit time can be significantly lower than a classical set-associative cache hit time (15-30% are reported). Without considering *optimistic execution*, the extra step induces a higher hit time for a set-associative cache than for a direct-mapped cache, however this may not be very significant [2].

In order to increase the clock frequency access to the cache may be pipelined. In the MIPS R4000 [8], the cache access has been divided into three independent stages (pipelining is implemented within the SRAM). Two pipeline stages are required to access the on-chip cache and a third stage is required to perform the tag check (see Figure 3A); the instruction tag is checked in RF stage simultaneously to decode. As a result, the processor cycle time

[1] The cache hit time is the delay, as seen by the processor, required by the memory system to service a memory reference on a hit [4]

[2] 2% was reported by Hill [4]