PAPER

Randomized Caches for Power-Efficiency

Hans VANDIERENDONCK[†a)] ***and*** **Koen De BOSSCHERE**[†b)]**,** ***Nonmembers***

SUMMARY Embedded processors are used in numerous devices executing dedicated applications. This setting makes it worthwhile to optimize the processor to the application it executes, in order to increase its power-efficiency. This paper proposes to enhance direct mapped data caches with automatically tuned randomized set index functions to achieve that goal. We show how randomization functions can be automatically generated and compare them to traditional set-associative caches in terms of performance and energy consumption. A 16 kB randomized direct mapped cache consumes 22% less energy than a 2-way set-associative cache, while it is less than 3% slower. When the randomization function is made configurable (i.e., it can be adapted to the program), the additional reduction of conflicts outweighs the added complexity of the hardware, provided there is a sufficient amount of conflict misses.

key words: *embedded processors, data cache, conflict miss, randomization*

1. Introduction

Numerous devices like PDA's, mobile phones, printers or network routers contain embedded computing systems. Although these processors will be used in a large variety of applications, they all share the common concerns for performance, low power consumption and low cost. However, there is greater freedom in designing embedded computing systems, as hardware and software are usually shipped as a whole. These circumstances present clear opportunities to tune the micro-architecture to specific applications in order to achieve a better trade-off between performance, cost and power consumption [1], [9].

Caches take a large part in the overall power budget of a processor (e.g.: 43% in StrongARM-1 [7]). By customizing the cache, the power-efficiency of the processor can be increased. This paper discusses a technique to customize data caches to applications by randomizing the set index function of the cache. The set index function determines the set of the cache that can hold a specific cache block. Traditionally, the lowest-order bits of the block address are selected to index the cache. A randomized set index function computes a more complex function of the address bits in order to avoid excessive conflict misses [13]. We consider only functions that perform exclusive or's (XOR) on the address bits [17], [18]. Randomization can be used in conjunction with set-associative caches, but the highest performance gain is obtained in direct mapped caches [15]. In fact, the miss rate of a well randomized direct mapped cache is lower than the miss rate of a 2-way set-associative cache. Randomization is especially interesting to use in a power-constrained setting, as set-associative caches consume significantly more power than direct mapped caches (typically in the range of 30–50%). Hence, randomization allows one to obtain the same performance benefits as associativity, but without the increase in power consumption.

Because a huge number of XOR-based randomization functions exist, and the performance of these functions varies widely [18], we developed a design space exploration tool to automatically generate randomization functions. The tool works in two phases. In the first phase, it collects profiling information for the applications and in the second phase, it uses the profiling information to generate a conflict-free randomization function [17], [18].

Several scenarios are possible to implement randomization. First, one can select one function during the design of a processor and implement it. Secondly, one can construct a programmable randomization function that is, e.g., initialized at the start of an application. A third possibility is to dynamically adapt the randomization function to the application. In this paper, the first two cases are investigated. The third case is left for future work.

The remainder of this paper is structured as follows. Section 2 discusses related work. Section 3 explains the design exploration methodology for randomization functions and Sect. 4 applies the tool to evaluate the power-efficiency of randomization using benchmarks taken from the Mediabench and MiBench suites. Section 5 summarizes the paper and Appendix details the design space exploration algorithm.

2. Related Work

Because microprocessors spend an extremely large fraction of their energy in the caches, many researchers have studied architectural techniques to lower active cache energy. The filter cache is a small cache that sits in front of the level 1 data cache [6]. Because it is

Manuscript received February 10, 2003.
Manuscript revised May 16, 2003.

[†]The authors are with the Department of Electronics and Information Systems, Ghent University, Belgium.
a) E-mail: hvdieren@elis.rug.ac.be
b) E-mail: kdb@elis.rug.ac.be