Power-Efficient Trace Caches*

J. S. Hu, N. Vijaykrishnan, M. Kandemir and M. J. Irwin
Dept. of Computer Science and Engineering
The Pennsylvania State University
{jhu, vijay, kandemir, mji}@cse.psu.edu

Abstract

This paper exploits the drawbacks of wasting power when accessing the instruction cache that stores only static sequence of instructions. Although trace cache is first introduced to catch the dynamic characteristics of instructions in execution, conventional trace cache (CTC) does increase the power consumption in fetch unit. A Sequential Trace Cache (STC) has been investigated for its power efficiency in this paper.

1. Introduction

Caching of dynamic sequences of instructions in a trace cache can avoid fetching unused instructions in cache lines, and has the potential to improve the power-efficiency over using a traditional instruction cache (I-cache). A trace cache stores traces, which can contain noncontiguous basic blocks and include several taken branches. Since traces are built at the instruction retiring stage, trace building does not affect the critical path of the instruction pipeline. As the conventional trace cache [1] (CTC) was proposed to improve the fetch width and performance, both trace cache and instruction cache are accessed simultaneously. When the required instructions are found in the trace cache, instructions fetched from the I-cache will be discarded, and accessing I-cache actually wastes power. In this paper, we investigate the Sequential Trace Cache [2] (STC) as a power efficient alternative of the traditional trace cache.

2. Sequential Trace Cache

Since trace cache captures the dynamic characteristics of instructions in execution, it has the potential to reduce the power consumption in the fetch unit. In our study, a trace is a dynamic sequence of instructions that contains at most sixteen instructions and at most three branches, depending on which one is first met. A trap instruction or a NOP instruction encountered cancels current trace building. Further, a new trace building will be terminated when an indirect branch or a return instruction is encountered.

The main purpose of designing the STC is to avoid the simultaneous access to trace cache and I-cache. In contrast to the conventional trace cache, the sequential trace cache and I-cache work in a sequential mode. At any fetch cycle, either the trace cache is accessed or the instruction cache is accessed. In the proposed architecture, the trace cache is placed between the CPU and L1 instruction cache. The L1 instruction cache is accessed only when there is a miss in trace cache. An additional machine flag bit – access mode flag is used to indicate whether the fetching address is sent to trace cache or instruction cache.

The trace building strategy is also sequential. Trace cache lookup is disabled during the building of a new trace. Only after a new trace building has been finished or trace fetching has been finished, the access mode flag will be set to indicate to access the trace cache in next cycle.

3. Results

We use Wattch to collect the simulation results. Figure 1 shows the power implications of the conventional trace cache and the STC. Since this paper focuses on the power consumption in the instruction fetch stage, we only consider the power consumed in the instruction cache and trace cache and refer to their sum as the fetch power. In Figure 1, a 64KB, 128B line size, 2way I-cache is used as baseline machine (BS), and CTC-64: BS + a 8KB 2way CTC, STC-64: BS + a 8KB 2way STC, STC-64-4: 32KB I-cache + 32KB 4way STC, STC-128-3: 16KB I-cache + 48KB 3way STC, and STC-128-4: 8KB I-cache + 64KB 4way STC. All I-caches are 2way associative. Cache line size is 128B. It can be observed that the conventional trace cache increases the fetch power consumption by 5.9% on average (as compared to BS), for five SPEC2000 CINT benchmarks and three media benchmarks. In contrast, the STC-64 can reduce 24.9% fetch power consumption as compared to BS, while STC-128-4 can improve this reduction further to 42.8%.
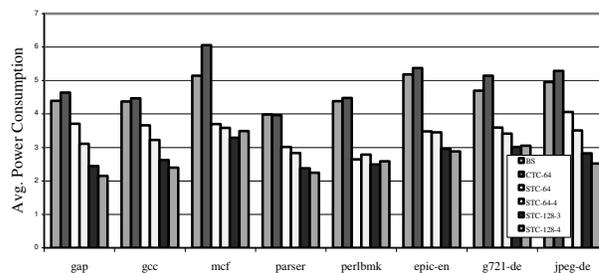

Figure 1. Power Cons. Of STC at Diff. Configurations.

References

[1] E. Rotenberg, S. Bennett, and J.E. Smith, Trace Cache: a Low Latency Approach to High Bandwidth Instruction Fetching, In Micro-96.
[2] J. Faistl, T. Jaracz, Trace Cache: Effect on Instruction Cache Miss Frequency, http://www.ece.cmu.edu/~ee742/proj_s98/faistl/index.html.