Microprocessors and Microsystems 38 (2014) 197–207

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

Linked instruction caches for enhancing power efficiency of embedded systems

Chang-Jung Ku, Ching-Wen Chen *, An Hsia, Chun-Lin Chen

Department of Information Engineering and Computer Science, Feng Chia University, Taichung City 40724, Taiwan

A R T I C L E I N F O

Article history:
Available online 15 February 2014

Keywords:
Instruction cache
Low power
Branch target buffer
Embedded systems

A B S T R A C T

The power consumed by memory systems accounts for 45% of the total power consumed by an embedded system, and the power consumed during a memory access is 10 times higher than during a cache access. Thus, increasing the cache hit rate can effectively reduce the power consumption of the memory system and improve system performance. In this study, we increased the cache hit rate and reduced the cache-access power consumption by developing a new cache architecture known as a single linked cache (SLC) that stores frequently executed instructions. SLC has the features of low power consumption and low access delay, similar to a direct mapping cache, and a high cache hit rate similar to a two way-set associative cache by adding a new link field. In addition, we developed another design known as a multiple linked caches (MLC) to further reduce the power consumption during each cache access and avoid unnecessary cache accesses when the requested data is absent from the cache. In MLC, the linked cache is split into several small linked caches that store frequently executed instructions to reduce the power consumption during each access. To avoid unnecessary cache accesses when a requested instruction is not in the linked caches, the addresses of the frequently executed blocks are recorded in the branch target buffer (BTB). By consulting the BTB, a processor can access the memory to obtain the requested instruction directly if the instruction is not in the cache. In the simulation results, our method performed better than selective compression, traditional cache, and filter cache in terms of the cache hit rate, power consumption, and execution time.

1. Introduction

Embedded systems are essentially application-specific microcomputers. Embedded system products are used widely nowadays, such as mobile phones, GPS receivers, and MP3 players. Most embedded systems are designed as mobile devices or are embedded in other devices where the power supply is mainly through batteries. Therefore, in addition to performance and cost considerations, power consumption is a significant factor. Thus, it is important to design a power-saving embedded system for prolonging the battery life of such products.

According to previous studies, memory system accesses account for about 45% [6,11,21–23] of the total power consumed by embedded systems. Thus, if the power consumed by the memory system were decreased, the total power consumed by a system would decrease significantly. According to [8], the power consumed for each instance of memory access is 8–10 times that consumed for each instance of cache access. Therefore, if the number of instances of memory access were to decrease, the power consumption of an embedded system would improve greatly.

In previous studies, many researchers have aimed to reduce the power consumption of memory systems to deliver power savings [1–5,7–8,10,15–18,21,24–29]. We categorize these works into three types: (1) adding a small cache between the CPU and the L1 cache to reduce the access power [2,15–16,24–25]. Previous authors [15] have stated that although the additional small cache can reduce memory system power consumption by 40–50%, it has a higher miss rate because it is too small to capture the locality of the executed programs. (2) Increasing the degree of cache associativity [10,21] to reduce the number of conflict misses also increases the number of tag comparisons, which increases the power consumption. Consequently, the power consumed for each instance of cache access increases as the degree of cache associativity increases. (3) Using a compression cache to reduce the number of memory accesses. [1,3–5,7–8,17–18,26–29]. However, this method incurs extra power and performance penalties when decompressing the instructions fetched from the compression cache. In addition, the addresses of the compressed instructions

* Corresponding author. Tel.: +886 424517250x3728.
E-mail addresses: p9522050@fcu.edu.tw (C.-J. Ku), chingwen@fcu.edu.tw (C.-W. Chen), p9943324@fcu.edu.tw (A. Hsia), m9806590@fcu.edu.tw (C.-L. Chen).

http://dx.doi.org/10.1016/j.micpro.2014.01.006
0141-9331/© 2014 Elsevier B.V. All rights reserved.

differ from the original instructions so it needs to translate an address before accessing the compression cache, which leads to extra power consumption.

According to the above analysis, the following problems should be considered when designing a power-aware cache-memory hierarchy: the cache hit rate, power consumption during one cache access, and avoidance of address translation. In this paper, we propose a new cache architecture called single linked cache (SLC), which addresses these problems to deliver power savings. First, a reference table is used to store the frequently executed instructions and increase the hit rate. We also provide a mapping relationship between the addresses and the frequently executed instructions in the reference table so a processor can fetch the requested instructions from the table without address translation. Second, we reduce the power consumption when accessing the reference table by using a low associative strategy in the reference table, while a mechanism of entry linkage is used to solve conflict misses due to low associativity. Finally, although frequently executed instructions comprise 10–20% of the total instructions in a program, only part of the frequently executed instructions may be used during a short period of time, i.e., not all frequently executed instructions are used during a short period. Thus, to further reduce the power consumption during a single access to the frequently executed instruction reference table, we designed a multiple linked caches (MLC) to reduce the access power. In addition, to avoid unnecessary cache accesses when a requested instruction is not in the linked caches, the addresses of the frequently executed blocks are also recorded in the branch target buffer (BTB). By consulting BTB, a processor can access the memory to obtain the requested instruction directly if the instruction is not in the cache, which avoids unnecessary cache access.

This paper is organized as follows. Section 2 reviews previous related works. Section 3 explains how the proposed SLC and MLC work. Section 4 presents the experimental results. In Section 5, we conclude this paper.

2. Related works

Previous studies of power-saving embedded systems have focused on reducing the power consumption of the processors and the memory system interface, mainly by reducing the memory access times and the power consumption of the memory bus. In addition to saving power, reducing the number of memory accesses can also reduce the execution time. Therefore, Benini and Yoshida proposed selective compression to reduce the number of memory accesses [4,29]. Benini [29] proposed a method that compresses frequently executed instructions and packages multiple compressed instructions into a virtual instruction. When a processor fetches a virtual instruction from the cache, the processor can decompress it to obtain multiple instructions and reduce the number of memory accesses. In [4], Yoshida proposed a method for storing frequently executed instructions in the memory without packaging. When a compressed instruction is fetched, the bus switches from its original 32-bit transmission to 8-bit transmission to reduce the power consumption.

These studies mainly reduced the number of memory accesses by compressing frequently executed instructions. However, to decompress the compressed instructions, the system must maintain a frequently executed instruction reference table that stores the original frequently executed instructions. When a virtual instruction or a compressed instruction is fetched, it can be decompressed to obtain multiple instructions or a single uncompressed instruction by looking up the reference table. Due to instruction compression, the address of an instruction in the memory is different from the instruction address issued by the processor. Therefore, the system needs a look-ahead table (LAT) to record the mapping relationship between the address of a compressed instruction in the memory and the address of an original instruction. Thus, there are overheads when accessing the LAT before fetching an instruction from the memory.

Other researchers [2,15–16,24–25] have suggested the design where a large cache is replaced by several small caches to save power when accessing the cache because the power consumption when accessing a large cache is considerably larger than that when accessing a small one. In these approaches, researchers added a small cache between the processor and the cache, which is known as a filter cache [15]. When a processor sends an address request, the filter cache is accessed first with lower power requirements to fetch the requested instruction if the access is a hit in the filter cache. If a miss occurs in the filter cache, the cache is accessed. Utilizing a filter cache can reduce the power consumption of accesses but it does not improve the access time because there is not much difference between the access time with a large cache and a small cache. The limited size of the filter cache also results in significant numbers of conflict misses, which degrade the system performance. Besides, Kim [14] added a small cache in the memory system called LPT-cache for capturing the most cache accesses to reduce power consumption by utilizing LPT-cache. They store the basic blocks of instructions into LPT-cache instead of cache lines in the traditional mapping way. In addition, the related information of the stored basic blocks is recorded in the branch target buffer, which is offered to the processor to fetch needed instructions from LPT-cache.

Zhang [30] pointed out that the utilization of the sets in a cache is non-uniform to result in some sets are seldom used to decrease the cache hit rates. Thus, Zhang proposed a new cache architecture called Efficient Cache to modify decoders in caches to balance the utilization of each set to improve cache hit rates and reduce the power consumption of memory system.

3. Proposed method

In this chapter, we introduce our novel system known as a linked cache that reduces the power consumption of the memory system. The proposed linked cache can increase the cache hit rate and avoid address translation, while also reducing the power consumption. In addition, a direct mapping strategy is used in the proposed linked cache to reduce the access power while a link field is added to a cache block to reduce conflict misses.

To further reduce the power consumption, we designed multiple linked caches where the linked cache is divided into several parts to reduce the access power. To avoid unnecessary cache accesses when the requested instructions are not in the linked caches, the addresses of the frequently executed instruction blocks are recorded in the branch target buffer (BTB). By consulting BTB, a processor can access the memory to obtain the requested instruction directly if the requested instruction is not in the cache, which avoids unnecessary cache access. Section 3.1 explains the design of the single linked cache (SLC) and Section 3.2 describes the design of multiple linked caches (MLC).

3.1. Design of the single linked cache (SLC)

In this section, we explain the design of our proposed SLC. To increase the cache hit rates and avoid address translation before fetching an instruction, we analyzed trace files to store the instructions of the most executed addresses in the linked cache using direct mapping and linking. Unlike previous studies [3] that selected the most executed instructions as the frequently executed instructions and that used cost address translation before accessing these frequently executed instructions, we collected the corresponding