Microprocessors and Microsystems 38 (2014) 451–457

Contents lists available at ScienceDirect

Microprocessors and Microsystems

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

Design of write merging and read prefetching buffer in DRAM controller for embedded processor

Chen Zhao, Kuizhi Mei [*], Nanning Zheng

Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University, Xi'an, Shaanxi, China

ARTICLE INFO

Article history:
Available online 12 April 2014

Keywords:
Write merging
Read prefetching
DRAM controller
Embedded processor

ABSTRACT

Write merging and read prefetching are effective methods for improving processor performance, and they are mainly used in processors for desktop or server. As embedded system requires more powerful microprocessor, how to improve the performance of embedded processor is worthy of concern. This paper presents the architecture of write merging and read prefetching buffer in DRAM controller for embedded processor. The evaluation model is constructed, and the result demonstrates that the proposed method can reduce cache miss penalty dramatically. Additionally, the design of DRAM controller with write merging and read prefetching buffer is implemented and verified on FPGA platform, it can reduce CPI by 19.6% on average. Moreover, the RTL module of presented design is synthesized by Design Compiler, and synthesis result shows that hardware cost of proposed architecture is relatively small compared to performance amelioration.

1. Introduction

Microprocessor is widely used in embedded systems; many embedded applications are memory intensive, such as multimedia applications, and image processing. In most embedded systems, DRAM is acting as the main memory. For DRAM operation model, a row must be activated first, in order to access data, and read or write operation is performed after a specified delay. When the access is completed, a precharge operation has to be performed. This results in long memory access latencies. How to tolerate the speed gap between cache and main memory, which is also called memory wall [1], is significant for microprocessor. For example, LEON2 as an open source version of LEON processors, which are 32-bit high reliable embedded processors provided by Aeroflex Gaisler [2], is used in our embedded vision microprocessor. For those memory intensive tasks, frequent memory access affects the performance of LEON2 adversely.

Prefetching is one of the several effective approaches that can be used to tolerate large memory latency, and it could improve performance if prefetch requirements are accurate and prefetched data is ready early enough. If prefetching engine in its cache hierarchy issues the prefetch request, this is the processor-side prefetch [3–6]. Memory-side prefetching is an alternative, where the engine that prefetches data for processor is in the main memory system [7–11]. Prefetching is always important and receives a lot attention. In the past, many prefetching methods have been proposed, such as tagged prefetching, stride prefetching, Markov prefetcher, and correlation prefetching. These methods are evaluated and compared using a standard simulation framework in [12]. In recent study, [13] incorporates dynamic feedback into the design of prefetcher, [14] proposed the row buffer caches for prefetching in die-stacked DRAMs. Besides academic research, data prefetching is also employed in many off the shelf processors today. Such as IBM POWER system processors, data prefetching is adopted not only between cache and main memory but also between the different levels of cache [15,16]. NVIDIA even integrates prefetching engine in the North Bridge chip [17].

Unlike prefetching, write merging attracts less attention, but it is also an effective method to hide memory access latencies. Write merging combines multiple writes within the same row in DRAM, which is essential for write through cache. Even for write back cache, when dirty cache line is replaced, if the write buffer is empty, all the data can be written to the write merging buffer. Besides, multi-words writes are more efficient than single word for DRAM after one row is activated. Ref. [18] discussed several issues in write buffer design, such as buffer depth, width, and retirement policy. A write merging buffer for embedded processor which employ write through cache is implemented, flip-flops array is used for parallel column address compare [19]. Write merging is

* Corresponding author.
E-mail addresses: chenzhao@mail.xjtu.edu.cn (C. Zhao), meikuizhi@mail.xjtu.edu.cn (K. Mei).

http://dx.doi.org/10.1016/j.micpro.2014.03.010
0141-9331/© 2014 Elsevier B.V. All rights reserved.

popular in today's processor, the Sun Niagara processor and Intel Core i7, among many others, use this method.

Write merging and prefetching are explored in depth previously, and they are also employed in powerful processor. But most of the previous researches have focused on processors for desktop or server applications, SPEC CPU benchmarks [20] are used to evaluate proposed methods. Although many architecture features of embedded processors originate from high performance desktop processors, processors used for embedded applications and desktop or server differs significantly. Ref. [21] provides clear evidence that SPEC benchmarks may be not suitable to evaluate or compare embedded systems. In addition, most of previous approaches proposed for high-end processors are complex, paying attention to only one approach, either write merging or prefetching, and some methods need compiler's intervention. Ref. [22] presents a scheme for reducing energy consumption of SDRAM access in embedded system by using cache-like prefetching buffer and combining writes to the same DRAM row, and the scheme is evaluated using MiBench applications [23], which is a benchmark set designed for presenting workloads of embedded applications, however, their simulation was accomplished without hardware implementation, apart from that, coherence problem caused by prefetching buffer on write request is also not illustrated in [22]. In this paper, we present a structure of write merging and read prefetching buffer in DRAM controller for embedded processor. Unlike write buffer designed for write back cache in [22], the write buffer presented in this paper is mainly for write through cache, and it is also compatible with write back cache. The proposed architecture in this paper is cache-like, and its control logic is not complex, which is appropriate to be adopted in memory controller of embedded processor.

The rest of this paper is organized as follows. First, the architecture of write merging and read prefetching buffer is introduced in Section 2, and the solution of coherence problem is also described in this section. Then, Section 3 details the evaluation methodology, results as well as its discussion. Next, Section 4 presents the hardware implementation, shows the prototype verification on FPGA platform and effect of proposed method for improving performance. Finally, Section 5 gives the conclusion and future work.

2. Proposed approach

The typical memory subsystem of embedded microprocessor is shown in Fig. 1(a). The baseline configuration consists of a processor, a bus and a memory controller. If cache access misses, the processor has to stall for many cycles waiting for fetching the required data from main memory. The architecture presented in this paper is shown in Fig. 1(b); write merging and read prefetching buffer are inserted between CPU bus and scheduler of memory controller to reduce memory access delay. Additionally, write merging buffer is beneficial to improve the bandwidth utilization of DRAM.

2.1. Write merging buffer

The write merging buffer is a cache like structure. Fig. 2 shows the write merging buffer (WMB) architecture with two entries. An entry includes a buffer line for a DRAM row, a tag, a valid, and a counter (ct in Fig. 2). The buffer line includes data part and offset address part, and data part is used to store data in the same row of DRAM, address part is used to store the corresponding column address. The tag bits part determines the bank and row address of DRAM, the valid bit indicates the storage line is either empty or valid, and the counter records the number of stored data.

WMB is fully associative which is small and easy to be implemented. Like the map relationship between cache and memory, this full associativity means that a row of DRAM can be mapped into any entry of WMB. For a write request, the address is partitioned into two parts, one is tag address corresponding to bank and row address of DRAM, another is offset address corresponding to column address of DRAM, and the control flow looks like Fig. 3. The tag address from bus is compared to every valid tag, which means write hit if there is one valid tag equal to tag address. If write hit and the matched line is not full, the offset address and data from bus should be written into the first empty location in the matched line, the counter indicates which one is first empty. If write hit, but the matched line is full, this line would be replaced and all the data in this line is written to DRAM, then the offset address and data of write request is written to the first location of this line, the counter is set to one. It means write miss if no valid tag is equal to tag address. If write miss and there is one or more empty lines, the data and offset address from bus is written to the first location in the empty line with minimum number. If write miss, but there is not empty line, one line should be chosen to be replaced. Several replacement algorithms are often adopted, such as FIFO, Random, LRU, but these methods cannot ensure the storage line with as much as possible data being written to DRAM. Therefore, MVD (the entry with most valid data is replaced) is chosen as the replacement algorithm. Based on the mechanism mentioned above, it can insure that the map between write merging buffer line and row of DRAM is one to one.

2.2. Read prefetching buffer

Read prefetching buffer (RPB) is also a cache like structure; it is consisted of entries, an entry includes tag bits, valid bit, and data

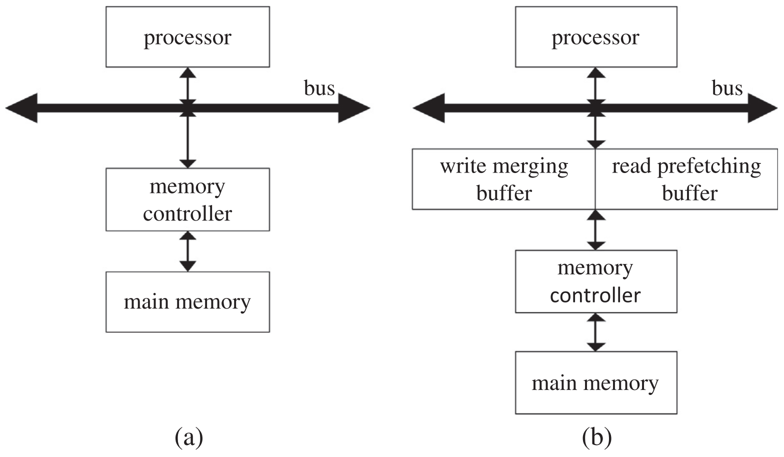

Fig. 1. Architecture of memory subsystem.