Design of an Intelligent Data Cache with Replacement Policy

B. Shameedha Begum, National Institute of Technology Tiruchirappalli, Tiruchirappalli, India
N. Ramasubramanian, National Institute of Technology Tiruchirappalli, Tiruchirappalli, India

ABSTRACT

Embedded systems are designed for a variety of applications ranging from Hard Real Time applications to mobile computing, which demands various types of cache designs for better performance. Since real-time applications place stringent requirements on performance, the role of the cache subsystem assumes significance. Reconfigurable caches meet performance requirements under this context. Existing reconfigurable caches tend to use associativity and size for maximizing cache performance. This article proposes a novel approach of a reconfigurable and intelligent data cache (L1) based on replacement algorithms. An intelligent embedded data cache and a dynamic reconfigurable intelligent embedded data cache have been implemented using Verilog 2001 and tested for cache performance. Data collected by enabling the cache with two different replacement strategies have shown that the hit rate improves by 40% when compared to LRU and 21% when compared to MRU for sequential applications which will significantly improve performance of embedded real time application.

KEYWORDS
Cache Memory, Dynamic Reconfiguration, Embedded Data Cache, Embedded Systems, Replacement Policies, Verilog

1. INTRODUCTION

Applications that execute on devices other than the conventional computers such as automobiles, home appliances, devices used in factory automation or mobile phones may be broadly termed as Embedded Applications. Several programming languages like VHDL, Verilog BSV, etc., and Design tools like Xilinx ISE or Aldec HDL are available to meet design challenges and optimize them (Crisu, 1999). Embedded Systems are characterized by operations that must be performed to meet strict deadlines as well as power and energy requirements.

In order to build an ideal embedded system, the designer has to optimize energy and power requirement, the speed of execution of an embedded processor, and the overall cost of the system. Memory is the main bottleneck of any computing system. Since the speed of the main memory unit is limited by electronic and packaging constraints, nowadays desktops, as well as server CPUs, have at least 3 different caches. First, an instruction cache is used to increase the speed of the instruction fetch process, next data cache is used to reduce the process of fetching and storing the data, and finally a Translation Lookaside Buffer (TLB). Generally, the data cache is organized as a Multilevel Cache Memories such as L1, L2 etc. Cache memory is used in all modern computing systems to reduce the overall average time. It is the smaller and faster semiconductor memory which stores frequently referred information. Therefore, this cache memory mechanism is widely popular in embedded hardware architecture such as IC, core base ASIC design and it is well suited for embedded real-time systems (Jacob, n.d.; Hennessy & Patterson, 2011).
As the cache is small in size, it gets filled up sooner. When there is no room for the new entry into the cache, then the replacement policy will decide the victim to be purged. There are various replacement policies available in the literature. In order to achieve good cache performance, the parameters such as Hit Rate, Miss Rate, Miss Penalty, and Hit cost should be considered. If a cache contains a valid data, then it is Hit and Miss otherwise. When there is a cache miss, the cache control hardware decides which block should be removed to create room for the new block that contains the referenced word. The gather of guidelines for making this rule constitutes the replacement algorithm.

Performance of cache can be optimized by increasing cache size, associativity, adjust block size or choose a suitable replacement policy that is ideal for each embedded application. Ruchi Rastogi et al. (2010) have proposed to change block size and associativity of cache that matches with application behavior. But we feel there is room to improve their effort in order to gain further performance improvement. In our work, we propose new IED (Intelligent Embedded Data) cache and DRIED (Dynamic Reconfigurable Intelligent Embedded Data) cache model. In IED cache we have implemented two replacement policies LRU and MRU. LRU discards the least recently used data items and MRU evicts the most recently used items from the block. Both the algorithms cover a wide variety of embedded applications. In IED we can statically choose between either replacements policies at the beginning of execution. In DRIED cache model we reused the idea of IED cache and change the replacement policies by observing the behavior of each replacement policy to the application being run.

This paper is organized as follows: Section 2 describes an overview of existing systems. In Section 3 the proposed reconfigurable cache architecture is described. In Section 4, the implementation of the cache is explained. Section 5 shows the evaluation results. Finally, we conclude the paper in section 6.

2. OVERVIEW OF EXISTING SYSTEMS

Embedded Systems, it is the craft of picking and planning the correct combination of equipment and programming parts to attain outline objectives like speed and efficiency. The embedded systems which have hard real-time constraints are used for focused categories of applications. Performance of processor and cache are of immense importance to meet those time constraints. Also, cache contributes to performance optimization of embedded processors. Hence there is a need to optimize cache performance.

The impact of the level-2 cache memory organization on performance and power consumption are extensively studied by A. Asaduzzaman et al. (2009). The different architecture that involves L2 cache, either private or shared or mixed with the homogeneous multi-core system has been considered for evaluation. The simulation results show that when adding more cores will have a negligible impact on mean interconnection delay and that favors distributed L2 cache. It is better to include an L2 cache when there are more than 2 cores with associativity 8 or less and the cache sizes should be 64Kb. A mechanism to reduce the refreshing energy eDRAM L2 cache is proposed by Alejandro Valero et al. (2015) by checking whether the blocks are getting refreshed or not, by applying the reuse-based refresh policy that combines with Most Recently Used-Tour (MRUT) replacement algorithm. Evaluations in this work were mainly focused on the L2 cache of single core processors. The effects of combining the code compression and dynamic cache reconfiguration are being extensively studied by Hadi Hajimir et al. (2012). The compression aware dynamic cache reconfiguration (DCR) finds best energy optimal cache configuration for each application through an exhaustive search of all possible cache configurations and the energy consumption can be analyzed by considering the hit and miss rates and this can be applied to compressed code. This scheme is applied to L2 cache to tune it for energy efficiency. The compression technique is confined to the instruction cache and code compression was performed offline during the evaluation.

In order to solve the problem of cache interference in multithreaded processors, a dynamic cache management system is proposed by Alex Settle et al. in (2006) that minimizes the inter and intra-thread
MAC and Routing Integration in Wireless Sensor Networks
www.igi-global.com/chapter/mac-routing-integration-wireless-sensor/65673?camid=4v1a

Parallel Access Control Model in Cross-Domain Grid Computing Environment
www.igi-global.com/article/parallel-access-control-model-in-cross-domain-grid-computing-environment/193621?camid=4v1a