Available online at www.sciencedirect.com

ScienceDirect

Procedia Technology 25 (2016) 544 – 551

Procedia Technology

Global Colloquium in Recent Advancement and Effectual Researches in Engineering, Science and Technology (RAEREST 2016)

Improved Architecture for Tag Matching in Cache memory Coded with Error Correcting Codes

Mary Francy Joseph[a], Anith Mohan[b*]

[a]*PG scholar,College of engineering,Munnar 685612 , India*
[b]*Assistant professor,College of engineering,Munnar 685612,India*

Abstract

Cache memories serve as accelerators to improve the performance of modern microprocessors. Caches are vulnerable to soft errors because of technology scaling. So it is important to provide protection mechanisms against soft errors. Tag comparison is critical in cache memories to keep data integrity and high hit ratio. Error correcting codes (ECC) are used to enhance reliability of memory structures. The previous solution for cache access is to decode each cache way to detect and correct errors. In the proposed architecture ECC delay is moved to the non-critical path of the process by directly comparing the retrieved tag with the incoming new information which is encoded as well, thus reducing circuit complexity. For the efficient computation of hamming distance, butterfly weight accumulator is proposed to reduce latency and complexity further. The proposed architecture checks whether the incoming data matches the stored data. The proposed architecture reduces the latency and hardware complexity compared with the most recent implementation.

Keywords: Error correcting codes; hamming distance ; cache memory; soft errors

1. Introduction

Cache memories are utilized for the low-latency access for data and instruction memory. Caches store instructions and data that are frequently used during operations. Microprocessors will first check whether a piece of information is in a cache.

doi:10.1016/j.protcy.2016.08.143

For that the address of the information in the cache is compared to all the cache tags that might contain that address. If the information is found there then there is a cache hit otherwise there will be cache miss. In the case of cache miss the piece of information will be found from the main memory, but it will simply take longer [1]. Caches must be designed in such a way to have as low latency as possible, or the components will be disqualified from serving as accelerators and the overall performance of the whole system will be deteriorated.

The reliability of cache memory is affected by soft errors. Soft errors can corrupt the instruction and data in the cache memory. Soft errors are not reproducible and can cause malfunctions in the hardware. Errors in cache memory can easily circulate into the processor registers and other memory elements. Soft errors in digital circuits are mitigated using large number of techniques [3]. To combat against soft errors cache memories use error protection mechanisms such as parity codes and SEC-DED (single-bit error correction and double-bit error detection) codes.

The caches, in addition to data and instructions store tag field to identify each cache line. Caches have to operate with low latency, the use of error correcting codes (ECC) is challenging because complex error protection schemes for tag bits can increase the latency [2].

Data comparison circuits are widely used in cache memories to perform tag matching. Data comparison is usually in the critical path of the pipeline stage that is devised to increase the performance of the system. The flow of the succeeding operations is determined by the output of the comparison. Protection using ECC increases the latency.

In cache, the tag directory is protected with ECC. When an access is made to the cache, the cache tag directory is accessed first. After retrieving the tag the encoded data go through ECC decoders and ECC correction logic before it is compared with tag field of the incoming address [4]. In this case, critical path is too long in cache systems designed for high speed access.

Another solution for matching problem is encode and compare method. In this method the incoming data is encoded and compared with the retrieved data that is encoded [6]. Complex decoding from the critical path is eliminated. The method does not look whether the retrieved data is exactly same as the incoming tag. Instead it looks whether the retrieved data is in the error correctable range of the codeword of the incoming data [7].

In this paper a new technique to efficiently implement tag matching in cache memory is presented. Here the comparison of the tag is done in parallel with the encoding process to generate parity. The latency and complexity for tag matching is reduced considerably.

2. Related Work

Tag arrays are typically protected with error correcting codes. Soft errors in tag bits cause pseudo hits and pseudo misses. Pseudo hit is a hit that is actually a miss in the absence of soft error is called pseudo hit. Pseudo hit lead to use the incorrectly matched data and is likely to propagate into other elements of the system. Pseudo miss is a miss that is actually a hit when there is no error [5]. In this case data is fetched from main memory producing performance degradation. To protect data and tag bits error correcting codes are used. When the tag array is protected with error correcting codes the latency is increased due to ECC logic. Many techniques have been presented to perform tag matching.

2.1 Decode and compare method

The tag array is protected with error correcting codes. Here the encoded data is read from the tag array first. As in Fig. 1 the encoded data go through ECC decoders and ECC correction logic before it is compared with tag of the incoming address.