T-EMU: Generic Cache Device Model Manual

© Terma GmbH, Germany, 2015. Proprietary and intellectual rights of Terma GmbH, Germany are involved in the subject-matter of this material and all manufacturing, reproduction, use, disclosure and sales rights pertaining to such subject-matter are expressly reserved. This material is submitted for a specific purpose as agreed in writing, and the recipient by accepting this material agrees that this material will not be used, copied or reproduced in whole or in part, nor its content (or any part thereof) revealed in any manner or to any third party, except own staff, to meet the purpose for which it was submitted and subject to the terms of the written agreement.
1. Introduction

T-EMU supports the use of cache models. However, cache models, at least when they are non-statistical have a significant impact on performance, and therefore, normally cache models are not used when running the emulator.

For the cases where a cache model is needed, the generic cache model is likely to be useful (see limitations for when it is less useful). It is a highly configurable cache model and supports being used, both as Harvard style caches (separate I- and D-caches) and as a unified cache.

CAUTION: When connecting the generic cache model in the memory hierarchy, it will intercept every memory transaction, and disable the ATC for any fetched, read or written data. This means that performance is significantly impacted firstly due to the need to visit the memory system for every fetch, read and write, but also and especially in a system with an enabled MMU, in these systems, the CPU will need to do a VM table walk for every memory access, which is very costly in terms of performance. Note that these table walks may be optimised in the future.

2. Configuration

2.1. Arguments

- **size**: Unified cache size in bytes.
- **instrSize**: Instruction cache size in bytes.
- **dataSize**: Data cache size in bytes.
- **ways**: Number of ways in a unified cache (must be power of 2)
- **instrWays**: Number of ways in instruction cache (must be power of 2)
- **dataWays**: Number of ways in data cache (must be power of 2)
2.2. Properties

The following properties are used for configuring the cache model and to connect the model in the object graph.

- **nextLevel**: Memory access interface reference for next memory level. Normally a memory space or a second (next level cache).
- **icacheCtrl**: Optional interface reference for a instruction cache controller object.
- **dcacheCtrl**: Optional interface reference for a data cache controller object.
- **instr.replacementPolicy**: Replacement policy used when fetching instructions. Set to 0 = NONE (or directly mapped / 1-way set associative cache). 1 = LRU, 2 = LRR and 3 = RND. Automatically set to 0 when ways is set to 1.
- **data.replacementPolicy**: Replacement policy used when accessing data. Set to 0 = NONE (or directly mapped / 1-way set associative cache). 1 = LRU, 2 = LRR and 3 = RND. Automatically set to 0 when ways is set to 1.
- **isSplitCache**: Cache is split and has separate instruction and data caches.
- **isWriteBack**: Cache is write-back cache, not supported at the moment.
- **isWriteAllocate**: Set to non-zero to have the cache allocate a line in case of a write miss. Set to zero to avoid line allocation.
- **fetchPenalty**: Cost for fetching from a cached line.
readPenalty
writePenalty
wordSize
instr.sets
instr.ways
instr.lineSize
data.sets
data.ways
data.lineSize

3. Properties

The generic cache model contains the following counters that can be inspected to get an idea of hit and miss-rates.

fetchHits
fetchMisses
readHits
readMisses
writeHits
writeMisses

4. Limitations

- The cache does not emulate write-back penalties for write-back caches at present. This means that the evict functions will behave as the invalidate functions.

- Number of ways must be a power of 2. That means that 1- 2- and 4- way set associative caches are fine, but 3-way set associative caches are not emulated by the generic cache model.