Doc. no: TERMA/SPD/63/TEMU/DEV/GENCACHE PUBLIC
Rev: 1.1
Date: 2016-09-06
Approved By:

TERMA

TEMU

Generic Cache Device Model Manual

Mattias Holm

Version 1.1, 2016-09-06

© Terma GmbH, Germany, 2015-2016. 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.

Table of Contents

Table 1. Record of Changes

Rev	Date	Author	Note
1.1	2016-09-16	MH	General improvements.
1.0	2015-09-18	MH	Initial version.

1. Introduction

TEMU supports the use of cache models. However, cache models, at least when they are non-statistical have a significant impact on performance, and therefor, 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 Harward 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.

The cache model will handle memory accesses with the TEMU_MT_CACHEABLE flag set. This flag can be set when mapping in a device (e.g. RAM or ROM).

2. Configuration === 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)

lineSize

Line size for unified cache

dataLineSize

Line size for data cache

instrLineSize

Line size for instruction cache

wordSize

Size of a word in bytes (defaults to 4)

separate

Set to 1 to turn the cache model to separate I- and D-caches. Set to 0 to make the cache a unified cache. This option affects the interpretation of the size, ways and lineSize arguments (see above).

2.1. Interfaces

The following interfaces can be used to connect the generic cache model:

PreAccessIface

A MemAccessIface that receives memory access events before they reach the target device.

PostAccessIface

A MemAccessIface that handles memory access events after they reach the target device.

2.2. Properties

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

preTransaction

Memory access interface reference for next pre-access handler.

postTransaction

Memory access interface reference for next post-access handler.

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

Cost for reading from a cached line.

writePenalty

Cost for writing to a cached line.

wordSize

Word size for cache (defaults to 4, do not modify unless connecting to 64-bit processor architectures).

instr.sets

Number of sets in the instruction cache.

instr.ways

Number of ways in the instruction cache.

instr.lineSize

Instruction line size in bytes.

data.sets

Number of sets in the data cache.

data.ways

Number of ways in the data cache.

data.lineSize

The use and/or disclosure,etc. of the contents of this document (or any part thereof) is subject to the restrictions referenced on the front page.

Data line size in bytes.

3. Properties

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

fetchHits

Number of cache hits when fetching instructions.

fetchMisses

Number of cache misses when fetching instructions.

readHits

Number of cache hits when reading data.

readMisses

Number of cache misses when reading data.

writeHits

Number of cache hits when writing data.

writeMisses

Number of cache misses when writing data.

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.