VciXcache

1) Functional Description

This VCI initiator is a generic cache controller, fully compliant with the VCI advanced protocol. Thanks to a normalized interface (source:trunk/soclib/soclib/communication/xcache/caba/source/include/xcache_signals.h?), this blocking cache controller can be used by several 32 bits RISC processors (such as Mips R3000, Sparc V8, or PPC 405). It contains two separated instruction and data caches, sharing the same VCI interface.

- The VCI ADDRESS and DATA fields must have 32 bits, and the VCI ERROR field must have 1 bit.
- The number of lines must be a power of 2, and cannot be larger than 1024.
- The number of words must be a power of 2, and cannot be larger than 32.

In order to garanty the memory consistency, this component does NOT start a new VCI transaction until the previous transaction is completed. Therefore, it does not use the VCI PKTID and TRDID fields.

Instruction Cache

- The Instruction cache is direct mapping and read-only.
- It uses the Mapping Table to support uncached segments.
- In case of read MISS, or read uncached, the processor is stalled until the missing instruction is available.
- The only VCI transaction generated by the Instruction cache is a read burst corresponding to a missing cache line.

Data Cache

- The Data cache is direct mapping.
- The write policy is WRITE-THROUGH (the data is immediately written in memory, and the cache is updated only in case of HIT).
- The Data cache contains a write buffer (8 words), and builds a burst when there are successive write requests with incrementing addresses.
- It uses the Mapping Table to support uncached segments.
- The Data Cache supports the following requests : Read, Write, Linked load, and Store Conditional
- The Data cache accepts a line invalidate command.
- Three types of VCI transactions can be generated by the data cache:
    - read burst of fixed length, corresponding to a cached read MISS,
    - one word transaction, corresponding to an uncached read, a linked load, or a store conditional.
    - write burst of variable length,
- The processor is stalled in case of cached read MISS, in case of uncached read, or in case of write, if the write buffer is full.

2) Component definition & usage

source:trunk/soclib/soclib/module/internal_component/vci_xcache/caba/metadata/vci_xcache.sd

```
Uses( 'vci_xcache', **vci_parameters )
```

3) CABA Implementation

- interface : source:trunk/soclib/soclib/module/internal_component/vci_xcache/caba/source/include/vci_xcache.h
- implementation : source:trunk/soclib/soclib/module/internal_component/vci_xcache/caba/source/src/vci_xcache.cpp

CABA Constructor parameters

```
VciXCache(
    sc_module_name insname,
    const soclib::common::MappingTable &mt,
    const soclib::common::IntTab &index,
    size_t icache_lines,
    size_t icache_words,
    size_t dcache_lines,
    size_t dcache_words );
```

CABA Ports

- sc_in<bool> **p_resetn** : Global system reset
- sc_in<bool> **p_clk** : Global system clock
- soclib::caba::ICacheCachePort **p_icache** : Icache interface
- soclib::caba::DCacheCachePort **p_dcache** : Dcache interface
- soclib::caba::VciInitiator<vci_param> **p_vci** : The VCI port

4) TLM-T Implementation

In order to increase the simulation speed, the processor ISS is directly wrapped by the VciXcache TLM-T model. Therefore, the ISS type is actually a template parameter for the VciXcache component.

TLM-T sources

- interface : source:trunk/soclib/soclib/module/internal_component/vci_xcache/tlmt/source/include/vci_xcache.h
- implementation : source:trunk/soclib/soclib/module/internal_component/vci_xcache/tlmt/source/src/vci_xcache.cpp

TLM-T Constructor parameters

TLM-T Ports