

VciRam

Functional Description

This VCI target is an embedded SRAM controller. This hardware component handles independent memory segments. Each segment is defined by a base address and a size (number of bytes). Both the base and the size parameters must be multiple of 4. The segments allocated to a given instance of this component are defined in the [Mapping Table](#).

Each segment is implemented as a dynamically allocated array in the constructor.

A MultiRam will initialize its segments from a binary if an !ElfLoader? is attached to it.

Component definition

[source:trunk/soclib/soclib/module/internal_component/vci_ram/caba/metadata/vci_ram.sd?](#)

Usage

See [SoclibCc/VciParameters](#)

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

CABA Implementation

- interface : [source:trunk/soclib/soclib/module/internal_component/vci_ram/caba/source/include/vci_ram.h?](#)
- implementation :
[source:trunk/soclib/soclib/module/internal_component/vci_ram/caba/source/src/vci_ram.cpp?](#)

TLM-T Implementation

Constructor parameters

- Uninitialized VciRam

```
VciRam(  
    sc_module_name name,                // Instance name  
    const soclib::common::IntTab &index, // Target index  
    const soclib::common::MappingTable &mt) // Mapping Table
```

- Elf-Initialized VciRam

You may load a binary file, by creating a loader:

```
soclib::common::ElfLoader loader( "a.out" );  
VciMultiRam(  
    sc_module_name name,                // Instance name  
    const soclib::common::IntTab &index, // Target index  
    const soclib::common::MappingTable &mt, // Mapping Table  
    soclib::common::ElfLoader &loader); // Loader
```

On reset, any loadable segment in ELF file will be reloaded .

Ports

- `sc_in<bool> p_resetn` : hardware reset
- `sc_in<bool> p_clk` : clock
- `soclib::common::VciTarget<vci_param> p_vci` : The VCI port