

VciSimpleRam

1) Functional Description

This VCI target is an embedded SRAM controller. It is actually a simplified version of the VciRam component, and provide the same services : it handles one or several 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 must be defined in the [Mapping Table](#). The segments are implemented as dynamically allocated arrays in the constructor.

This component supports an - optionnal - latency parameter, defining the RAM access latency. A L value for this parameter corresponds to a (L+1) cycles latency.

As the VciRam component, the VciSimpleRam component initializes its segments from a ELF binary if a [Loader](#) is attached to it.

2) Component definition & usage

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

See [SoclibCc/VciParameters](#)

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

3) CABA Implementation

CABA sources

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

CABA Constructor parameters

- Uninitialized VciSimpleRam

```
VciSimpleRam(  
    sc_module_name name,                // Instance name  
    const soclib::common::IntTab &index, // Target index  
    const soclib::common::MappingTable &mt, // Mapping Table  
    const uint32_t latency);             // Latency (Optionnal argument)
```

- Elf-Initialized VciSimpleRam

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

```
soclib::common::Loader loader( "a.out" );  
VciSimpleRam(  
    sc_module_name name,                // Instance name  
    const soclib::common::IntTab &index, // Target index
```

```

const soclib::common::MappingTable &mt, // Mapping Table
soclib::common::Loader &loader,         // Loader
const uint32_t latency);                // Latency (Optionnal argument)

```

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

CABA Ports

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

4) TLM-DT Implementation

TLM-DT sources

- interface :
source:trunk/soclib/soclib/module/internal_component/vci_simple_ram/tlmdt/source/include/vci_simple_ram.h
- implementation :
source:trunk/soclib/soclib/module/internal_component/vci_simple_ram/tlmdt/source/src/vci_simple_ram.cpp

TLM-DT Constructor parameters

```

VciSimpleRam(
    sc_module_name name, // Instance name
    const soclib::common::IntTab &index, // Target index
    const soclib::common::MappingTable &mt, // Mapping Table
    soclib::common::Loader &loader); // Loader

```

TLM-DT Ports

- `soclib::tlmdt::VciTarget<vci_param> p_vci` : The VCI port