base-components

Generated by Doxygen 1.8.17

Chapter 1

Main Page

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

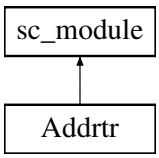
Class Documentation

4.1 Addrtr Class Reference

A Addrtr component that can add addrtr to a virtual platform project to manage the various transactions.

`#include <addrtr.h>`

Inheritance diagram for Addrtr:



Public Member Functions

- **Addrtr** (const sc_core::sc_module_name &nm)
- **Addrtr** (const Addrtr &)=delete

Public Attributes

- tlm_utils::simple_target_socket< Addrtr > **front_socket**
- tlm_utils::simple_initiator_socket< Addrtr > **back_socket**
- cci::cci_param< uint64_t > **offset**

4.1.1 Detailed Description

A Addrtr component that can add addrtr to a virtual platform project to manage the various transactions.

This component models a addrtr. It has a single multi-target socket so any other component with an initiator socket can connect to this component. It behaves as follows:

- Manages exclusive accesses, adding this addrtr as a 'hop' in the exclusive access extension (see GreenSocs/libgsutils).
- Manages connections to multiple initiators and targets with the method `add_initiator` and `add_target`.
- Allows to manage read and write transactions with `b_transport` and `transport_dbg` methods.
- Supports passing through DMI requests with the method `get_direct_mem_ptr`.
- Handles invalidation of multiple DMI pointers with the method `invalidate_direct_mem_ptr` which passes the invalidate back to *all* initiators.
- It checks for each transaction if the address is valid or not and returns an error if the address is invalid with the method `decode_address`.

The documentation for this class was generated from the following file:

- /home/thomas/Documents/GreenSocs/build-lib/base-components/include/greensocs/base-components/misc/addrtr.h

4.2 gs::CSVRow Class Reference

Public Member Functions

- std::string **operator[ ]** (std::size_t index) const
- std::size_t **size** () const
- void **readNextRow** (std::istream &str)

The documentation for this class was generated from the following file:

- /home/thomas/Documents/GreenSocs/build-lib/base-components/include/greensocs/base-components/loader.h

4.3 gs::Loader< BUSWIDTH >::elf_reader::elf32_traits Struct Reference

Public Types

- typedef Elf32_Ehdr **Elf_Ehdr**
- typedef Elf32_Phdr **Elf_Phdr**
- typedef Elf32_Shdr **Elf_Shdr**
- typedef Elf32_Sym **Elf_Sym**

Static Public Member Functions

- static Elf_Ehdr * **elf_getehdr** (Elf *elf)
- static Elf_Phdr * **elf_getphdr** (Elf *elf)
- static Elf_Shdr * **elf_getshdr** (Elf_Scn *scn)

The documentation for this struct was generated from the following file:

- /home/thomas/Documents/GreenSocs/build-lib/base-components/include/greensocs/base-components/loader.↩ h

4.4 gs::Loader< BUSWIDTH >::elf_reader::elf64_traits Struct Reference

Public Types

- typedef Elf64_Ehdr **Elf_Ehdr**
- typedef Elf64_Phdr **Elf_Phdr**
- typedef Elf64_Shdr **Elf_Shdr**
- typedef Elf64_Sym **Elf_Sym**

Static Public Member Functions

- static Elf_Ehdr * **elf_getehdr** (Elf *elf)
- static Elf_Phdr * **elf_getphdr** (Elf *elf)
- static Elf_Shdr * **elf_getshdr** (Elf_Scn *scn)

The documentation for this struct was generated from the following file:

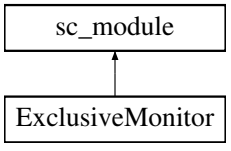
- /home/thomas/Documents/GreenSocs/build-lib/base-components/include/greensocs/base-components/loader.↩ h

4.5 ExclusiveMonitor Class Reference

ARM-like global exclusive monitor.

`#include <exclusive-monitor.h>`

Inheritance diagram for ExclusiveMonitor:



Public Member Functions

- **ExclusiveMonitor** (const sc_core::sc_module_name &name)
- **ExclusiveMonitor** (const ExclusiveMonitor &)=delete

Public Attributes

- tlm_utils::simple_target_socket< ExclusiveMonitor > **front_socket**
- tlm_utils::simple_initiator_socket< ExclusiveMonitor > **back_socket**

4.5.1 Detailed Description

ARM-like global exclusive monitor.

This component models an ARM-like global exclusive monitor. It connects in front of an target and monitors accesses to it. It behaves as follows:

- On an exclusive load, it internally marks the corresponding region as locked. The load is forwarded to the target.
- On an exclusive store to the same region, the region is unlocked, and the store is forwarded to the target.
- When an initiator perform an exclusive load while already owning a region, the region gets unlocked before the new one is locked.
- A regular store will unlock all intersecting regions
- If an exclusive store fails, that it, corresponds to a region which is not locked, or is locked by another initiator, or does not exactly match the store boundaries, the failure is reported into the TLM exclusive extension and the store is *not* forwarded to the target.
- DMI invalidation is performed when a region is locked.
- DMI requests are intercepted and modified accordingly to match the current locking state.
- DMI hints (the is_dmi_allowed() flag in transactions) is also intercepted and modified if necessary.

The documentation for this class was generated from the following file:

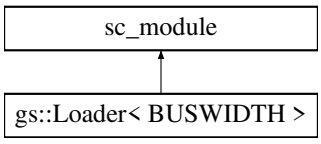
- /home/thomas/Documents/GreenSocs/build-lib/base-components/include/greensocs/base-components/misc/exclusive-monitor.h

4.6 gs::Loader< BUSWIDTH > Class Template Reference

A loader that cope with several formats.

`#include <loader.h>`

Inheritance diagram for gs::Loader< BUSWIDTH >:



Public Member Functions

- **Loader** (sc_core::sc_module_name name)
- **Loader** (sc_core::sc_module_name name, std::function< void(const uint8_t *data, uint64_t offset, uint64_t len)> _write)
- void file_load (std::string filename, uint64_t addr)

  This function reads a file into memory and can be used to load an image.
- void **csv_load** (std::string filename, uint64_t offset, std::string addr_str, std::string value_str, bool byte_swap)
- void **str_load** (std::string data, uint64_t addr)
- void **ptr_load** (uint8_t *data, uint64_t addr, uint64_t len)
- void **elf_load** (const std::string &path)

Public Attributes

- simple_initiator_socket_zero< Loader< BUSWIDTH > > **initiator_socket**

Protected Member Functions

- void **load** (std::string name)
- void **end_of_elaboration** ()

4.6.1 Detailed Description

template<unsigned int BUSWIDTH = 32>
class gs::Loader< BUSWIDTH >

A loader that cope with several formats.

4.6.2 Member Function Documentation

4.6.2.1 file_load()

```
template<unsigned int BUSWIDTH = 32>
void gs::Loader< BUSWIDTH >::file_load (
            std::string filename,
            uint64_t addr )  [inline]
```

This function reads a file into memory and can be used to load an image.

Parameters

filename	Name of the file
addr	the address where the memory file is to be read

Returns

size_t

The documentation for this class was generated from the following file:

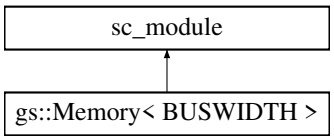
- /home/thomas/Documents/GreenSocs/build-lib/base-components/include/greensocs/base-components/loader.↩ h

4.7 gs::Memory< BUSWIDTH > Class Template Reference

A memory component that can add memory to a virtual platform project.

`#include <memory.h>`

Inheritance diagram for gs::Memory< BUSWIDTH >:



Public Member Functions

- **Memory** (sc_core::sc_module_name name, uint64_t _size=0)
- void **before_end_of_elaboration** ()
- **Memory** (const Memory &)=delete
- uint64_t size ()

  this function returns the size of the memory
- uint64_t base ()

  this function returns the base address of the memory

Public Attributes

- gs::Loader **load**
- tlm_utils::simple_target_socket< Memory, BUSWIDTH > **socket**
- cci::cci_param< bool > **p_rom**
- cci::cci_param< bool > **p_dmi**
- cci::cci_param< bool > **p_verbose**
- cci::cci_param< sc_core::sc_time > **p_latency**
- cci::cci_param< std::string > **p_mapfile**

Protected Member Functions

- virtual bool **get_direct_mem_ptr** (tlm::tlm_generic_payload &txn, tlm::tlm_dmi &dmi_data)
- virtual void **b_transport** (tlm::tlm_generic_payload &txn, sc_core::sc_time &delay)
- virtual unsigned int **transport_dbg** (tlm::tlm_generic_payload &txn)
- void **cci_ignore** (std::string name)
- void **read** (uint8_t *data, uint64_t offset, uint64_t len)
- void **write** (const uint8_t *data, uint64_t offset, uint64_t len)
- void map (std::string filename)

  This function maps a host file system file into the memory, such that the results of the memory will be maintained between runs. This can be useful for emulating a flash ram for instance NB the only way of having this called is via the configuration paramter mapfile.

4.7.1 Detailed Description

template<unsigned int BUSWIDTH = 32>
class gs::Memory< BUSWIDTH >

A memory component that can add memory to a virtual platform project.

This component models a memory. It has a simple target socket so any other component with an initiator socket can connect to this component. It behaves as follows:

- The memory does not manage time in any way
- It is only an LT model, and does not handle AT transactions
- It does not manage exclusive accesses
- You can manage the size of the memory during the initialization of the component
- Memory does not allocate individual "pages" but a single large block
- It supports DMI requests with the method `get_direct_mem_ptr`
- DMI invalidates are not issued.

4.7.2 Member Function Documentation

4.7.2.1 base()

```
template<unsigned int BUSWIDTH = 32>
uint64_t gs::Memory< BUSWIDTH >::base ( )  [inline]
```

this function returns the base address of the memory

Returns

the address of the memory of type uint64_t

4.7.2.2 map()

```
template<unsigned int BUSWIDTH = 32>
void gs::Memory< BUSWIDTH >::map (
             std::string filename )  [inline], [protected]
```

This function maps a host file system file into the memory, such that the results of the memory will be maintained between runs. This can be useful for emulating a flash ram for instance NB the only way of having this called is via the configuration paramter mapfile.

Parameters

filename	Name of the file

4.7.2.3 size()

```
template<unsigned int BUSWIDTH = 32>
uint64_t gs::Memory< BUSWIDTH >::size ( )  [inline]
```

this function returns the size of the memory

Returns

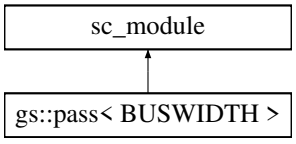
the size of the memory of type uint64_t

The documentation for this class was generated from the following file:

- /home/thomas/Documents/GreenSocs/build-lib/base-components/include/greensocs/base-components/memory.↩ h

4.8 gs::pass< BUSWIDTH > Class Template Reference

Inheritance diagram for gs::pass< BUSWIDTH >:



Public Member Functions

- **pass** (const sc_core::sc_module_name &nm, bool _verbose=false)
- **pass** (const pass &)=delete

Public Attributes

- initiator_socket_spying< pass< BUSWIDTH > > **initiator_socket**
- tlm_utils::simple_target_socket< pass< BUSWIDTH >, BUSWIDTH > **target_socket**
- cci::cci_param< bool > **p_verbose**

The documentation for this class was generated from the following file:

- /home/thomas/Documents/GreenSocs/build-lib/base-components/include/greensocs/base-components/connectors.↩ h

4.9 Path Class Reference

Path recording TLM extension.

4.9.1 Detailed Description

Path recording TLM extension.

extension

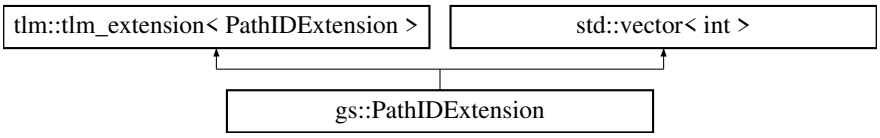
Embeds an ID field in the txn, which is populated as the network is traversed - see README.

The documentation for this class was generated from the following file:

- /home/thomas/Documents/GreenSocs/build-lib/base-components/include/greensocs/base-components/pathid↩ _extension.h

4.10 gs::PathIDExtension Class Reference

Inheritance diagram for gs::PathIDExtension:



Public Member Functions

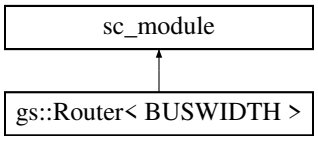
- **PathIDExtension** (const PathIDExtension &)=default
- virtual tlm_extension_base ∗ **clone** () const override
- virtual void **copy_from** (const tlm_extension_base &ext) override

The documentation for this class was generated from the following file:

- /home/thomas/Documents/GreenSocs/build-lib/base-components/include/greensocs/base-components/pathid↩ _extension.h

4.11 gs::Router< BUSWIDTH > Class Template Reference

Inheritance diagram for gs::Router< BUSWIDTH >:



Public Types

- typedef multi_passthrough_initiator_socket_spying< Router< BUSWIDTH > > **initiator_socket_type**

Public Member Functions

- **Router** (const sc_core::sc_module_name &nm)
- **Router** (const Router &)=delete
- void **add_target** (TargetSocket &t, const uint64_t address, uint64_t size, bool masked=true)
- virtual void **add_initiator** (InitiatorSocket &i)

Public Attributes

- initiator_socket_type **initiator_socket**
- tlm_utils::multi_passthrough_target_socket< Router< BUSWIDTH > > **target_socket**
- cci::cci_broker_handle **m_broker**
- cci::cci_param< bool > **thread_safe**

Protected Member Functions

- virtual void **before_end_of_elaboration** ()

The documentation for this class was generated from the following file:

- /home/thomas/Documents/GreenSocs/build-lib/base-components/include/greensocs/base-components/router.↩
  h

Index