qbox

Generated by Doxygen 1.8.13

Contents

Chapter 1

Main Page

Libqbox encapsulates QEMU in SystemC such that it can be instanced as a SystemC TLM-2.0 model.

1.1 GreenSocs Build and make system

1.2 How to build

This project may be built using cmake

```
cmake -B build;pushd build; make -j; popd
```

cmake may ask for your git.greensocs.com credentials (see below for advice about passwords)

1.2.1 cmake version

cmake version 3.14 or newer is required. This can be downloaded and used as follows

```
curl -L https://github.com/Kitware/CMake/releases/download/v3.20.0-rc4/cmake-3.20.0-rc4-linux-x86_64.tar.gz
       | tar -zxf -
./cmake-3.20.0-rc4-linux-x86_64/bin/cmake
```

1.2.2 details

This project uses CPM https://github.com/cpm-cmake/CPM.cmake in order to find, and/or download missing components. In order to find locally installed SystemC, you may use the standards SystemC environment variables: SYSTEMC_HOME and CCI_HOME. CPM will use the standard CMAKE find_package mechanism to find installed packages https://cmake.org/cmake/help/latest/command/find_package.↩ html To specify a specific package location use <package>_ROOT CPM will also search along the CMAKE↩ _MODULE_PATH

Sometimes it is convenient to have your own sources used, in this case, use the CPM_<package>_SOUR↩ CE_DIR. Hence you may wish to use your own copy of SystemC CCI "bash cmake -B build -DCPM_↩ SystemCCCI_SOURCE=/path/to/your/cci/source

```
It may also be convenient to have all the source files downloaded, you may do this by running
```bash
cmake -B build -DCPM_SOURCE_CACHE=`pwd`/Packages
```

This will populate the directory Packages Note that the cmake file system will automatically use the directory called Packages as source, if it exists.

NB, CMake holds a cache of compiled modules in ~/.cmake/ Sometimes this can confuse builds. If you seem to be picking up the wrong version of a module, then it may be in this cache. It is perfectly safe to delete it.
```

1.2.2.1 Common CMake options

CMAKE_INSTALL_PREFIX : Install directory for the package and binaries. CMAKE_BUILD_TYPE : DEBUG or RELEASE

The library assumes the use of C++14, and is compatible with SystemC versions from SystemC 2.3.1a.

For a reference docker please use the following script from the top level of the Virtual Platform:

```
curl --header 'PRIVATE-TOKEN: W1Z9U8S_5BUEX1_Y29iS'
       'https://git.greensocs.com/api/v4/projects/65/repository/files/docker_vp.sh/raw?ref=master' -o docker_vp.sh
chmod +x ./docker_vp.sh
./docker_vp.sh
> cmake -B build;cd build; make -j
```

1.2.2.2 passwords for git.greensocs.com

To avoid using passwords for git.greensocs.com please add a ssh key to your git account. You may also use a key-chain manager. As a last resort, the following script will populate ∼/.git-credentials with your username and password (in plain text)

```
git config --global credential.helper store
```

1.2.3 More documentation

More documentation, including doxygen generated API documentation can be found in the /docs directory.

1.2.4 LIBGSSYNC

The GreenSocs Synchronization library provides a number of different policies for synchronizing between an external simulator (typically QEMU) and SystemC.

These are based on a proposed standard means to handle the SystemC simulator. This library provides a backwards compatibility layer, but the patched version of SystemC will perform better.

1.2.5 The GreenSocs SystemC simple components library.

This includes simple models such as routers, memories and exclusive monitor. The components are "Loosely timed" only. They support DMI where appropriate, and make use of CCI for configuration.

It also has several unit tests for memory, router and exclusive monitor.

1.2.6 LIBGSUTILS

The GreenSocs basic utilities library contains utility functions for CCI, simple logging and test functions. It also includes some basic tlm port types

1.2.7 LIBQEMU-CXX

Libqemu-cxx encapsulates QEMU as a C++ object, such that it can be instanced (for instance) within a SystemC simulation framework.

1.2.8 Information about building and using the greensocs Qbox library

The greensocs Qbox library depends on the libraries : base-components, libgssync, libqemu-cxx, libgsutils, SystemC, RapidJSON, SystemCCI, Lua and GoogleTest.

1.2.9 Information about building and using the libqemu-cxx library

The libqemu-cxx library depends only on the libqemu library

1.2.10 Information about building and using the base-components library

The base-components library depends on the libraries : Libgsutls, SystemC, RapidJSON, SystemCCI, Lua and GoogleTest.

1.2.11 Information about building and using the libgssync library

The libgssync library depends on the libraries : base-components, libgsutils, SystemC, RapidJSON, SystemCCI, Lua and GoogleTest.

1.2.12 Information about building and using the libgsutils library

The libgsutils library depends on the libraries : SystemC, RapidJSON, SystemCCI, Lua and GoogleTest.

The GreenSocs CCI libraries allows two options for setting configuration parameters

`--gs_luafile <FILE.lua>` this option will read the lua file to set parameters.

`--param path.to.param=<value>` this option will allow individual parameters to be set.

NOTE, order is important, the last option on the command line to set a parameter will take preference.

This library includes a Configurable Broker (gs::ConfigurableBroker) which provides additional functionality. Each broker can be configured separately, and has a parameter itself for the configuration file to read. This is `lua_file`. Hence

```
--param path.to.module.lua_file="\"/host/path/to/lua/file""
```

Note that a string parameter must be quoted.

The lua file read by the ConfigurableBroker has relative paths - this means that in the example above the `path.to.module` portion of the absolute path should not appear in the (local) configuration file. (Hence changes in the hierarchy will not need changes to the configuration file).

1.2.13 Using yaml for configuration

If you would prefer to use yaml as a configuration language, `lyaml` provides a link. This can be downloaded from https://github.com/gvvaughan/lyaml

The following lua code will load "conf.yaml".

```
local lyaml   = require "lyaml"

function readAll(file)
    local f = assert(io.open(file, "rb"))
    local content = f:read("*all")
    f:close()
    return content
end

print "Loading conf.yaml"
yamldata=readAll("conf.yaml")
ytab=lyaml.load(yamldata)
for k,v in pairs(ytab) do
    _G[k]=v
end
yamldata=nil
ytab=nil
```

1.2.14 Instanciate Qemu

A QemuManager is required in order to instantiate a Qemu instance. A QemuManager will hold, and maintain the instance until the end of execution. The QemuInstance can contain one or many CPU's and other devices. To create a new instance you can do this:

```
{c++}
    QemuInstanceManager m_inst_mgr;
```

then you can initialize it by providing the QemuInstance object with the QemuInstanceManager object which will call the `new_instance` method to create a new instance.

```
{c++}
    QemuInstance m_qemu_inst(m_inst_mgr.new_instance(QemuInstance::Target::AARCH64))
```

In order to add a CPU device to an instance they can be constructed as follows:

```
{c++}
    sc_core::sc_vector<QemuCpuArmCortexA53> m_cpus

    m_cpus("cpu", 32, [this] (const char *n, size_t i) { return new QemuCpuArmCortexA53(n, m_qemu_inst); })
```

You can change the CPUs to those listed below in the "CPU" section

Interrupt Controllers and others devices also need a QEMU instance and can be set up as follows:

```
{c++}
    QemuArmGicv3 m_gic("gic", m_qemu_inst);
    QemuUartPl011 m_uart("uart", m_qemu_inst)
```

1.2.15 The components of libqbox

1.2.15.1 CPU

The libqbox library supports several CPU architectures such as ARM and RISCV.

- In ARM architectures the library supports the cortex-a53 and the Neoverse-N1 which is based on the cortex-a76 architecture which itself derives from the cortex-a75/73/72.
- In RISCV architecture, the library manages only the riscv64.

1.2.15.2 IRQ-CTRL

The library also manages interrupts by providing :

- ARM GICv2
- ARM GICv3 which are Arm Generic Interrupt Controller.

Then :

- SiFive CLINT
- SiFive PLIC which are also Interrupt controller but for SiFive.

1.2.15.3 UART

Finally, it has 2 uarts:

- pl011 for ARM
- 16550 for more general use

1.2.15.4 PORTS

The library also provides socket initiators and targets for Qemu

1.2.16 The GreenSocs component library memory

The memory component allows you to add memory when creating an object of type `Memory("name",size)`.

The memory component consists of a simple target socket :`tlm_utils::simple_target_socket<Memory> socket`

1.2.17 The GreenSocs component library router

The router offers `add_target(socket, base_address, size)` as an API to add components into the address map for routing. (It is recommended that the addresses and size are CCI parameters).

It also allows to bind multiple initiators with `add_initiator(socket)` to send multiple transactions. So there is no need for the bind() method offered by sockets because the add_initiator method already takes care of that.

1.2.18 Functionality of the synchronization library

In addition the library contains utilities such as an thread safe event (async_event) and a real time speed limited for SystemC.

1.2.18.1 Suspend/Unsuspend interface

This patch adds four new basic functions to SystemC:

```
void sc_suspend_all(sc_simcontext* csc= sc_get_curr_simcontext())
void sc_unsuspend_all(sc_simcontext* csc= sc_get_curr_simcontext())
void sc_unsuspendable()
void sc_suspendable()
```

suspend_all/unsuspend_all : This pair of functions requests the kernel to 'atomically suspend' all processes (using the same semantics as the thread suspend() call). This is atomic in that the kernel will only suspend all the processes together, such that they can be suspended and unsuspended without any side effects. Calling suspend_all(), and subsiquently calling unsuspend_all() will have no effect on the suspended status of an individual process. A process may call suspend_all() followed by unsuspend_all, the calls should be 'paired', (multiple calls to either suspend_all() or unsuspend_all() will be ignored). Outside of the context of a process, it is the programmers responsibility to ensure that the calls are paired. As a consequence, multiple calls to suspend_all() may be made (within separate process, or from within sc_main). So long as there have been more calls to suspend_all() than to unsuspend_all(), the kernel will suspend all processes.

[note, this patch set does not add convenience functions, including those to find out if suspension has happened, these are expected to be layered ontop]

unsusbendable()/suspendable(): This pair of functions provides an 'opt-out' for specific process to the suspend↩ _all(). The consequence is that if there is a process that has opted out, the kernel will not be able to suspend_all (as it would no longer be atomic). These functions can only be called from within a process. A process should only call suspendable/unsuspendable in pairs (multiple calls to either will be ignored). *Note that the default is that a process is marked as suspendable.*

Use cases: *1 : Save and Restore* For Save and Restore, the expectation is that when a save is requested, 'suspend_all' will be called. If there are models that are in an unsuspendable state, the entire simulation will be allowed to continue until such a time that there are no unsuspendable processes.

2 : External sync When an external model injects events into a SystemC model (for instance, using an 'async_↩ request_update()'), time can drift between the two simulators. In order to maintain time, SystemC can be prevented from advancing by calling suspend_all(). If there are process in an unsuspendable state (for instance, processing on behalf of the external model), then the simulation will be allowed to continue. NOTE, an event injected into the kernel by an async_request_update will cause the kernel to execute the associated update() function (leaving the suspended state). The update function should arrange to mark any processes that it requires as unsuspendable before the end of the current delta cycle, to ensure that they are scheduled.

1.2.19 Using the ConfigurableBroker

The broker will self register in the SystemC CCI hierarchy. All brokers have a parameter `lua_file` which will be read and used to configure parameters held within the broker. This file is read at the *local* level, and paths are *relative* to the location where the ConfigurableBroker is instanced.

These brokers can be used as global brokers.

The `gs::ConfigurableBroker` can be instanced in 3 ways:

1. `ConfigurableBroker()` This will instance a 'Private broker' and will hide **ALL** parameters held within this broker.

   A local `lua_file` can be read and will set parameters in the private broker. This can be prevented by passing 'false' as a construction parameter (`ConfigurableBroker(false)`).

2. `ConfigurableBroker({{"key1","value1"},{"key2","value2")...})` This will instance a broker that sets and hides the listed keys. All other keys are passed through (exported). Hence the broker is 'invisible' for parameters that are not listed. This is specifically useful for structural parameters.

   It is also possible to instance a 'pass through' broker using `ConfigurationBroker({})`. This is useful to provide a *local* configuration broker than can, for instance, read a local configuration file.

   A local `lua_file` can be read and will set parameters in the private broker (exported or not). This can be prevented by passing 'false' as a construction parameter (`ConfigurableBroker(false)`). The `lua_file` will be read **AFTER** the construction key-value list and hence can be used to over-right default values in the code.

3. `ConfigurableBroker(argc, argv)` This will instance a broker that is typically a global broker. The argc/argv values should come from the command line. The command line will be parsed to find:

   `> -p, --param path.to.param=<value>` this option will allow individual parameters to be set.

   `> -l, --gs_luafile <FILE.lua>` this option will read the lua file to set parameters. Similar functionality can be achieved using –param lua_file="<FILE.lua>".

   A `{{key,value}}` list can also be provided, otherwise it is assumed to be empty. Such a list will set parameter values within this broker. These values will be read and used **BEFORE** the command line is read.

   Finally **AFTER** the command line is read, if the `lua_file` parameter has been set, the configuration file that it indicates will also be read. This can be prevented by passing 'false' as a construction parameter (`ConfigurableBroker(argc, argv, false)`). The `lua_file` will be read **AFTER** the construction key-value list, and after the command like, so it can be used to over-right default values in either.

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 QemuInstanceDmiManager::DmiRegion Class Reference

a DMI region

`#include <dmi-manager.h>`

Public Types

- using **Key** = uintptr_t
- using **Ptr** = std::shared_ptr< DmiRegion >

Public Member Functions

- **DmiRegion** (const tlm::tlm_dmi &info, qemu::LibQemu &inst)
- uint64_t **get_size** () const
- qemu::MemoryRegion **get_mr** ()
- Key **get_key** () const
- bool **is_valid** () const
- void **invalidate** ()

Static Public Member Functions

- static Key **key_from_tlm_dmi** (const tlm::tlm_dmi &info)

4.1.1 Detailed Description

a DMI region

Represent a DMI region with a size and an host pointer. It also embeds the QEMU memory region mapping to this host pointer. Note that it does not have start and end addresses as it is totally address space agnostic. Two initiators with two different views of the address space can map the same DMI region.

Note: The get_key method is used to index the map in which the regions are stored. Currently, we use the host memory address itself to index the map. This makes a strong assumption on the fact that two consecutive DMI region requests for the same region will return the same host address. This is not clearly stated in the TLM-2.0 standard but is quite reasonable to assume.

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/dmi-manager.h

4.2 QemuInstanceDmiManager::DmiRegionAlias Class Reference

An alias to a DMI region.

`#include <dmi-manager.h>`

Public Member Functions

- **DmiRegionAlias** (DmiRegion::Ptr region, const tlm::tlm_dmi &info, qemu::LibQemu &inst)
- uint64_t **get_start** () const
- uint64_t **get_end** () const
- uint64_t **get_size** () const
- qemu::MemoryRegion **get_alias_mr** () const
- bool is_valid () const

  Return true if the alias and its underlying DMI region are valid.
- void invalidate_region ()

  Invalidate the underlying DMI region.
- void set_installed ()

  Mark the alias as mapped onto QEMU root MR.
- bool is_installed () const

  Return true if the alias is mapped onto QEMU root MR.

4.2.1 Detailed Description

An alias to a DMI region.

An object of this class represents an alias to a DMI region a CPU can map on its own address space. Contrary to a DmiRegion, it has a start and an end address as it it requested from the point of view of an initiator's address map.

It embeds a shared pointer of the underlying DMI region. The DMI region get destroyed once all aliases referencing it have been destroyed.

4.2.2 Member Function Documentation

4.2.2.1 invalidate_region()

`void QemuInstanceDmiManager::DmiRegionAlias::invalidate_region ( )  [inline]`

Invalidate the underlying DMI region.

Note

Must be called with the DMI manager lock held

4.2.2.2 is_installed()

```
bool QemuInstanceDmiManager::DmiRegionAlias::is_installed ( ) const  [inline]
```

Return true if the alias is mapped onto QEMU root MR.

Note

Must be called with the DMI manager lock held

4.2.2.3 is_valid()

```
bool QemuInstanceDmiManager::DmiRegionAlias::is_valid ( ) const  [inline]
```

Return true if the alias and its underlying DMI region are valid.

Note

Must be called with the DMI manager lock held

4.2.2.4 set_installed()

```
void QemuInstanceDmiManager::DmiRegionAlias::set_installed ( )  [inline]
```

Mark the alias as mapped onto QEMU root MR.

Note

Must be called with the DMI manager lock held

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/dmi-manager.h

4.3 LockedQemuInstanceDmiManager Class Reference

A locked QemuInstanceDmiManager.

```
#include <dmi-manager.h>
```

Public Types

- using **DmiRegion** = QemuInstanceDmiManager::DmiRegion

Public Member Functions

- **LockedQemuInstanceDmiManager** (QemuInstanceDmiManager &inst)
- **LockedQemuInstanceDmiManager** (const LockedQemuInstanceDmiManager &)=delete
- **LockedQemuInstanceDmiManager** (LockedQemuInstanceDmiManager &&)=default
- QemuInstanceDmiManager::DmiRegionAlias get_new_region_alias (const tlm::tlm_dmi &info)

Protected Attributes

- QemuInstanceDmiManager & **m_inst**
- std::unique_lock< std::mutex > **m_lock**

4.3.1 Detailed Description

A locked QemuInstanceDmiManager.

This class is a wrapper around QemuInstanceDmiManager that ensure safe accesses to it. As long as an instance of this class is live, the underlying QemuInstanceDmiManager is locked. It gets unlocked once the object goes out of scope.

4.3.2 Member Function Documentation

4.3.2.1 get_new_region_alias()

```
QemuInstanceDmiManager::DmiRegionAlias LockedQemuInstanceDmiManager::get_new_region_alias (
            const tlm::tlm_dmi & info )  [inline]
```

See also

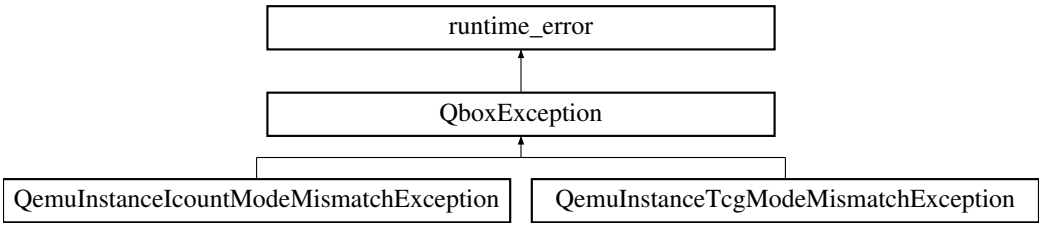
QemuInstanceDmiManager::get_new_region_alias

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/dmi-manager.h

4.4 QboxException Class Reference

Inheritance diagram for QboxException:



Public Member Functions

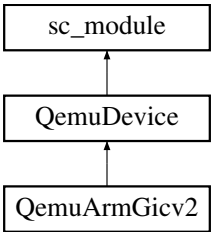
- **QboxException** (const char *what)

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/exceptions.h

4.5 QemuArmGicv2 Class Reference

Inheritance diagram for QemuArmGicv2:



Public Member Functions

- **QemuArmGicv2** (const sc_core::sc_module_name &name, QemuInstance &inst)
- void **before_end_of_elaboration** ()
- void **end_of_elaboration** ()

Public Attributes

- QemuArmGicv2m * **m_gicv2m**
- QemuTargetSocket **dist_iface**
- QemuTargetSocket **cpu_iface**
- QemuTargetSocket **virt_iface**
- QemuTargetSocket **vcpu_iface**
- QemuTargetSocket ::TlmTargetSocket **v2m_iface**
- sc_core::sc_vector< QemuTargetSignalSocket > **spi_in**
- sc_core::sc_vector< sc_core::sc_vector< QemuTargetSignalSocket > > **ppi_in**
- sc_core::sc_vector< QemuInitiatorSignalSocket > **irq_out**
- sc_core::sc_vector< QemuInitiatorSignalSocket > **fiq_out**
- sc_core::sc_vector< QemuInitiatorSignalSocket > **virq_out**
- sc_core::sc_vector< QemuInitiatorSignalSocket > **vfiq_out**
- sc_core::sc_vector< QemuInitiatorSignalSocket > **maintenance_out**

Static Public Attributes

- static const uint32_t **NUM_PPI** = 32

Protected Attributes

- cci::cci_param< unsigned int > **p_num_cpu**
- cci::cci_param< unsigned int > **p_num_spi**
- cci::cci_param< unsigned int > **p_revision**
- cci::cci_param< bool > **p_has_virt_extensions**
- cci::cci_param< bool > **p_has_security_extensions**
- cci::cci_param< unsigned int > **p_num_prio_bits**
- cci::cci_param< bool > **p_has_msi_support**

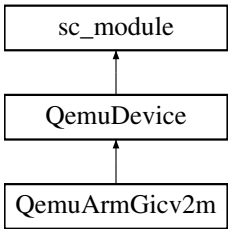
Additional Inherited Members

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/components/irq-ctrl/arm-gicv2.h

4.6 QemuArmGicv2m Class Reference

Inheritance diagram for QemuArmGicv2m:



Public Member Functions

- **QemuArmGicv2m** (const sc_core::sc_module_name &name, QemuInstance &inst)
- unsigned int **get_base_spi** ()
- unsigned int **get_num_spis** ()
- void **before_end_of_elaboration** ()
- void **end_of_elaboration** ()

Public Attributes

- sc_core::sc_vector< QemuInitiatorSignalSocket > **spi_out**
- QemuTargetSocket **iface**

Protected Attributes

- cci::cci_param< unsigned int > **p_base_spi**
- cci::cci_param< unsigned int > **p_num_spis**

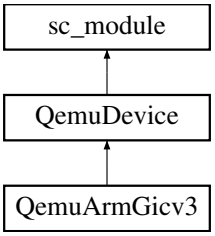
Additional Inherited Members

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/components/irq-ctrl/arm-gicv2.h

4.7 QemuArmGicv3 Class Reference

Inheritance diagram for QemuArmGicv3:



Public Member Functions

- **QemuArmGicv3** (const sc_core::sc_module_name &name, QemuInstance &inst)
- void **before_end_of_elaboration** ()
- void **end_of_elaboration** ()

Public Attributes

- QemuTargetSocket **dist_iface**
- sc_core::sc_vector< QemuTargetSocket<> > **redist_iface**
- sc_core::sc_vector< QemuTargetSignalSocket > **spi_in**
- sc_core::sc_vector< sc_core::sc_vector< QemuTargetSignalSocket > > **ppi_in**
- sc_core::sc_vector< QemuInitiatorSignalSocket > **irq_out**
- sc_core::sc_vector< QemuInitiatorSignalSocket > **fiq_out**
- sc_core::sc_vector< QemuInitiatorSignalSocket > **virq_out**
- sc_core::sc_vector< QemuInitiatorSignalSocket > **vfiq_out**

Static Public Attributes

- static const uint32_t **NUM_PPI** = 32

Protected Attributes

- cci::cci_param< unsigned int > **p_num_cpu**
- cci::cci_param< unsigned int > **p_num_spi**
- cci::cci_param< unsigned int > **p_revision**
- cci::cci_param< std::vector< unsigned int > > **p_redist_region**
- cci::cci_param< bool > **p_has_security_extensions**

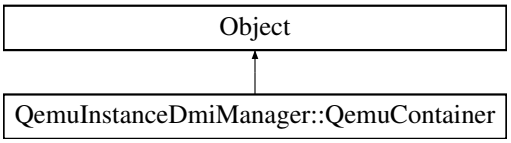
Additional Inherited Members

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/components/irq-ctrl/arm-gicv3.h

4.8 QemuInstanceDmiManager::QemuContainer Class Reference

Inheritance diagram for QemuInstanceDmiManager::QemuContainer:



Public Member Functions

- **QemuContainer** (const QemuContainer &o)=default
- **QemuContainer** (const Object &o)

Static Public Attributes

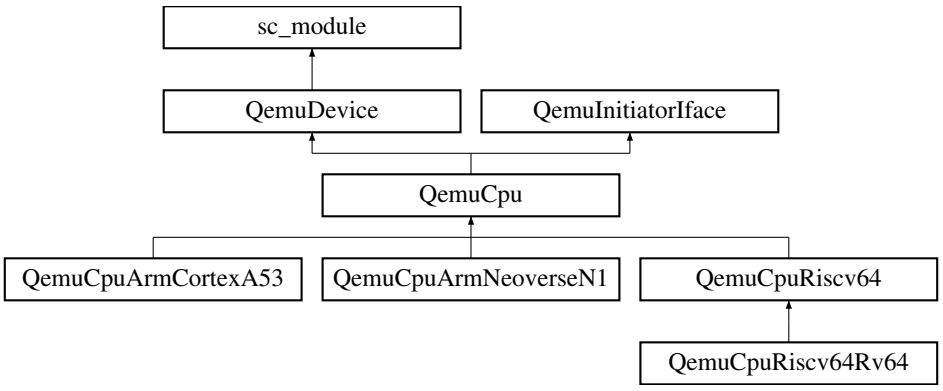
- static constexpr const char *const **TYPE** = "container"

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/dmi-manager.h

4.9 QemuCpu Class Reference

Inheritance diagram for QemuCpu:



Classes

- class QemuCpuHintTlmExtension

Public Member Functions

- **SC_HAS_PROCESS** (QemuCpu)
- **QemuCpu** (const sc_core::sc_module_name &name, QemuInstance &inst, const std::string &type_name)
- void **before_end_of_elaboration** () override
- virtual void **end_of_elaboration** () override
- virtual void **start_of_simulation** () override
- virtual void **initiator_customize_tlm_payload** (TlmPayload &payload) override
- virtual void **initiator_tidy_tlm_payload** (TlmPayload &payload) override
- virtual sc_core::sc_time **initiator_get_local_time** () override
- virtual void **initiator_set_local_time** (const sc_core::sc_time &t) override

Public Attributes

- cci::cci_param< bool > **p_icount**
- cci::cci_param< int > **p_icount_mips**
- cci::cci_param< unsigned int > **p_gdb_port**
- cci::cci_param< std::string > **p_sync_policy**
- QemuInitiatorSocket **socket**

Protected Member Functions

- void **create_quantum_keeper** ()
- void **set_qemu_instance_options** ()
- void **set_signaled** ()
- void **watch_external_ev** ()
- void **kick_cb** ()
- void **deadline_timer_cb** ()
- void **wait_for_work** ()
- void **rearm_deadline_timer** ()
- void **prepare_run_cpu** ()
- void **run_cpu_loop** ()
- void **sync_with_kernel** ()
- void **end_of_loop_cb** ()
- void **mainloop_thread_coroutine** ()

Protected Attributes

- gs::RunOnSysC **m_on_sysc**
- std::shared_ptr< qemu::Timer > **m_deadline_timer**
- bool **m_coroutines**
- qemu::Cpu **m_cpu**
- gs::async_event **m_qemu_kick_ev**
- sc_core::sc_event_or_list **m_external_ev**
- bool **m_signaled**
- std::mutex **m_signaled_lock**
- std::condition_variable **m_signaled_cond**
- int64_t **m_last_vclock**
- std::shared_ptr< gs::tlm_quantumkeeper_extended > **m_qk**
- QemuCpuHintTlmExtension **m_cpu_hint_ext**

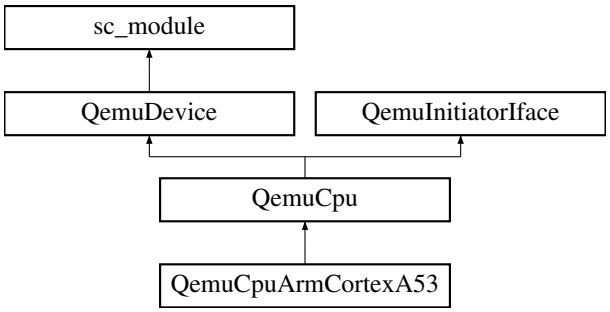
Additional Inherited Members

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/components/cpu/cpu.h

4.10 QemuCpuArmCortexA53 Class Reference

Inheritance diagram for QemuCpuArmCortexA53:



Public Member Functions

- **QemuCpuArmCortexA53** (sc_core::sc_module_name name, QemuInstance &inst)
- void **before_end_of_elaboration** () override
- void **end_of_elaboration** () override
- void **initiator_customize_tlm_payload** (TlmPayload &payload) override
- void **initiator_tidy_tlm_payload** (TlmPayload &payload) override

Public Attributes

- cci::cci_param< unsigned int > **p_mp_affinity**
- cci::cci_param< bool > **p_has_el2**
- cci::cci_param< bool > **p_has_el3**
- cci::cci_param< bool > **p_start_powered_off**
- cci::cci_param< std::string > **p_psci_conduit**
- cci::cci_param< uint64_t > **p_rvbar**
- QemuTargetSignalSocket **irq_in**
- QemuTargetSignalSocket **fiq_in**
- QemuTargetSignalSocket **virq_in**
- QemuTargetSignalSocket **vfiq_in**
- QemuInitiatorSignalSocket **irq_timer_phys_out**
- QemuInitiatorSignalSocket **irq_timer_virt_out**
- QemuInitiatorSignalSocket **irq_timer_hyp_out**
- QemuInitiatorSignalSocket **irq_timer_sec_out**

Static Public Attributes

- static constexpr qemu::Target **ARCH** = qemu::Target::AARCH64

Protected Member Functions

- int **get_psci_conduit_val** () const
- void **add_exclusive_ext** (TlmPayload &pl)

Static Protected Member Functions

- static uint64_t **extract_data_from_payload** (const TlmPayload &pl)

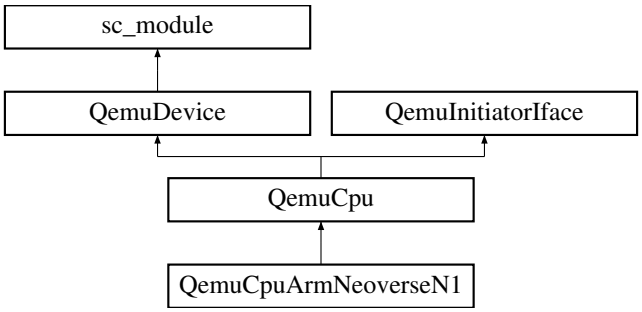
Additional Inherited Members

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/components/cpu/arm/cortex-a53.h

4.11 QemuCpuArmNeoverseN1 Class Reference

Inheritance diagram for QemuCpuArmNeoverseN1:



Public Member Functions

- **QemuCpuArmNeoverseN1** (sc_core::sc_module_name name, QemuInstance &inst)
- void **before_end_of_elaboration** () override
- void **end_of_elaboration** () override
- void **initiator_customize_tlm_payload** (TlmPayload &payload) override
- void **initiator_tidy_tlm_payload** (TlmPayload &payload) override

Public Attributes

- cci::cci_param< unsigned int > **p_mp_affinity**
- cci::cci_param< bool > **p_has_el2**
- cci::cci_param< bool > **p_has_el3**
- cci::cci_param< bool > **p_start_powered_off**
- cci::cci_param< std::string > **p_psci_conduit**
- cci::cci_param< uint64_t > **p_rvbar**
- QemuTargetSignalSocket **irq_in**
- QemuTargetSignalSocket **fiq_in**
- QemuTargetSignalSocket **virq_in**
- QemuTargetSignalSocket **vfiq_in**
- QemuInitiatorSignalSocket **irq_timer_phys_out**
- QemuInitiatorSignalSocket **irq_timer_virt_out**
- QemuInitiatorSignalSocket **irq_timer_hyp_out**
- QemuInitiatorSignalSocket **irq_timer_sec_out**

Static Public Attributes

- static constexpr qemu::Target **ARCH** = qemu::Target::AARCH64

Protected Member Functions

- int **get_psci_conduit_val** () const
- void **add_exclusive_ext** (TlmPayload &pl)

Static Protected Member Functions

- static uint64_t **extract_data_from_payload** (const TlmPayload &pl)

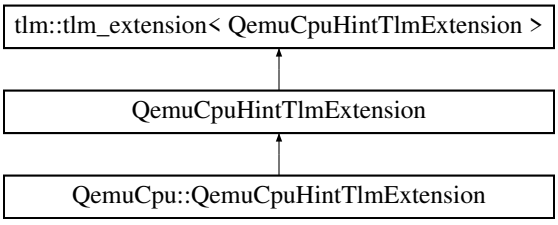
Additional Inherited Members

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/components/cpu/arm/neoverse-n1.h

4.12 QemuCpu::QemuCpuHintTlmExtension Class Reference

Inheritance diagram for QemuCpu::QemuCpuHintTlmExtension:



Public Member Functions

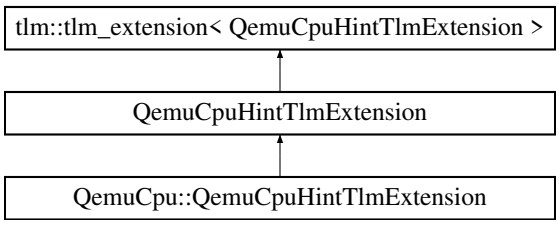
- void **free** () override

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/components/cpu/cpu.h

4.13 QemuCpuHintTlmExtension Class Reference

Inheritance diagram for QemuCpuHintTlmExtension:



Public Member Functions

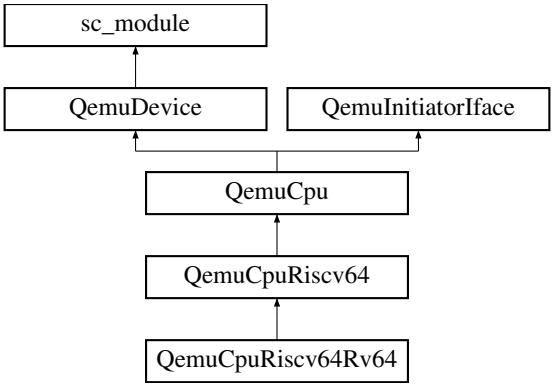
- **QemuCpuHintTlmExtension** (const QemuCpuHintTlmExtension &)=default
- **QemuCpuHintTlmExtension** (qemu::Cpu cpu)
- virtual tlm_extension_base * **clone** () const override
- virtual void **copy_from** (tlm_extension_base const &ext) override
- void **set_cpu** (qemu::Cpu cpu)
- qemu::Cpu **get_cpu** () const

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/tlm-extensions/qemu-cpu-hint.h

4.14 QemuCpuRiscv64 Class Reference

Inheritance diagram for QemuCpuRiscv64:



Public Member Functions

- **QemuCpuRiscv64** (const sc_core::sc_module_name &name, QemuInstance &inst, const char *model, uint64_t hartid)
- void **before_end_of_elaboration** ()

Protected Member Functions

- void **mip_update_cb** (uint32_t value)

Protected Attributes

- uint64_t **m_hartid**
- gs::async_event **m_irq_ev**

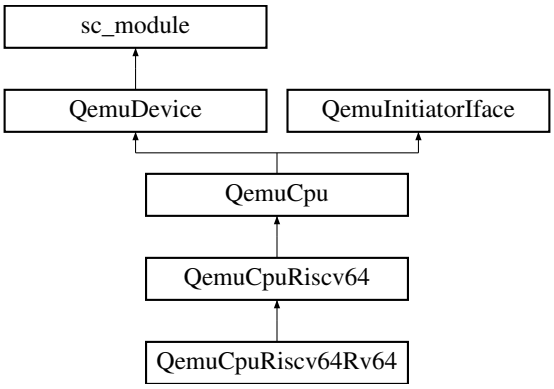
Additional Inherited Members

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/components/cpu/riscv64/riscv64.h

4.15 QemuCpuRiscv64Rv64 Class Reference

Inheritance diagram for QemuCpuRiscv64Rv64:



Public Member Functions

- **QemuCpuRiscv64Rv64** (const sc_core::sc_module_name &n, QemuInstance &inst, uint64_t hartid)

Additional Inherited Members

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

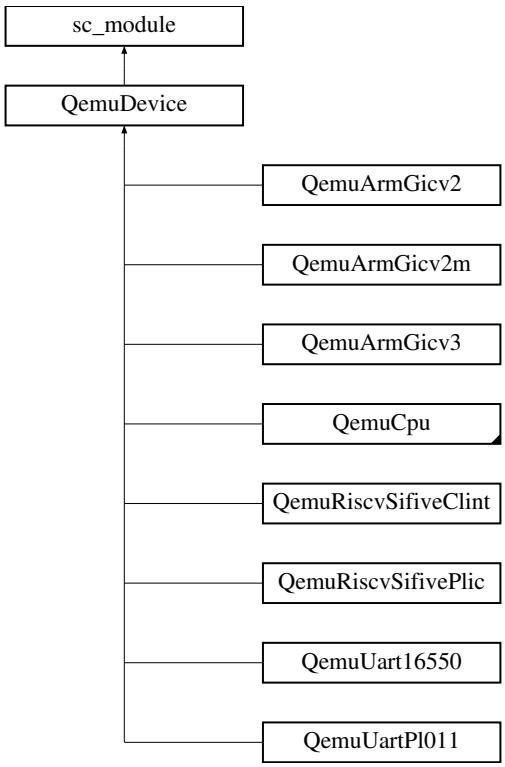
- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/components/cpu/riscv64/riscv64.h

4.16 QemuDevice Class Reference

QEMU device abstraction as a SystemC module.

`#include <device.h>`

Inheritance diagram for QemuDevice:



Public Member Functions

- QemuDevice (const sc_core::sc_module_name &name, QemuInstance &inst, const char *qom_type)
  Construct a QEMU device.
- virtual void **before_end_of_elaboration** () override
- virtual void **end_of_elaboration** () override
- const char * **get_qom_type** () const
- qemu::Device **get_qemu_dev** ()
- QemuInstance & **get_qemu_inst** ()

Protected Member Functions

- void **realize** ()

Protected Attributes

- QemuInstance & **m_inst**
- qemu::Device **m_dev**
- bool **m_realized** = false

4.16.1 Detailed Description

QEMU device abstraction as a SystemC module.

This class abstract a QEMU device as a SystemC module. It is constructed using the QEMU instance it will lie in, and the QOM type name corresponding to the device. This class is meant to be inherited from by children classes that implement a given device.

The elaboration flow is as follows:

- At construct time, nothing happen on the QEMU side.
- When before_end_of_elaboration is called, the QEMU object correponding to this component is created. Children classes should always call the parent method when overriding it. Usually, they start by calling it and then set the QEMU properties on the device.
- When end_of_elaboration is called, the device is realized. No more property can be set (unless particular cases such as some link properties) and the device can now be connected to busses and GPIO.

4.16.2 Constructor & Destructor Documentation

4.16.2.1 QemuDevice()

```
QemuDevice::QemuDevice (
            const sc_core::sc_module_name & name,
            QemuInstance & inst,
            const char * qom_type )  [inline]
```

Construct a QEMU device.

Parameters

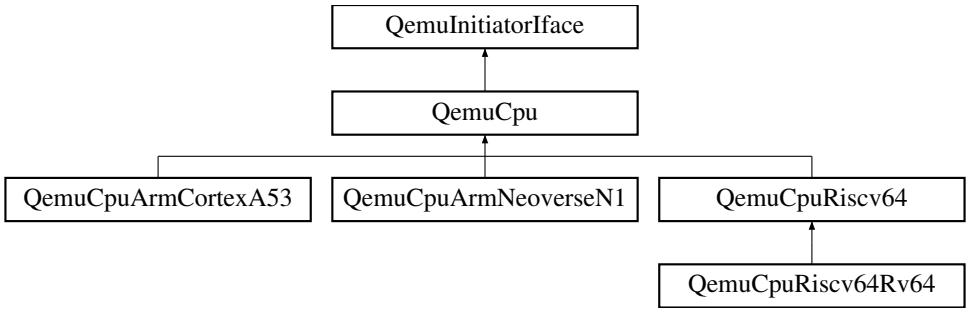
in	*name*	SystemC module name
in	*inst*	QEMU instance the device will be created in
in	*qom_type*	Device QOM type name

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/components/device.h

4.17 QemuInitiatorIface Class Reference

Inheritance diagram for QemuInitiatorIface:



Public Types

- using **TlmPayload** = tlm::tlm_generic_payload

Public Member Functions

- virtual void **initiator_customize_tlm_payload** (TlmPayload &payload)=0
- virtual void **initiator_tidy_tlm_payload** (TlmPayload &payload)=0
- virtual sc_core::sc_time **initiator_get_local_time** ()=0
- virtual void **initiator_set_local_time** (const sc_core::sc_time &)=0

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

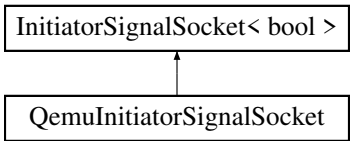
- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/ports/initiator.h

4.18 QemuInitiatorSignalSocket Class Reference

A QEMU output GPIO exposed as a InitiatorSignalSocket<bool>

`#include <initiator-signal-socket.h>`

Inheritance diagram for QemuInitiatorSignalSocket:



Public Member Functions

- **QemuInitiatorSignalSocket** (const char *name)
- void init (qemu::Device dev, int gpio_idx)

  Initialize this socket with a device and a GPIO index.
- void init_named (qemu::Device dev, const char *gpio_name, int gpio_idx)

  Initialize this socket with a device, a GPIO namespace, and a GPIO index.
- void init_sbd (qemu::SysBusDevice sbd, int gpio_idx)

  Initialize this socket with a QEMU SysBusDevice, and a GPIO index.

Protected Member Functions

- void **event_cb** (bool val)
- void **init_qemu_to_sysc_gpio_proxy** (qemu::Device &dev)
- void **init_internal** (qemu::Device &dev)

Protected Attributes

- qemu::Gpio **m_proxy**
- gs::RunOnSysC **m_on_sysc**
- QemuTargetSignalSocket * **m_qemu_remote** = nullptr

4.18.1 Detailed Description

A QEMU output GPIO exposed as a InitiatorSignalSocket<bool>

This class exposes an output GPIO of a QEMU device as a InitiatorSignalSocket<bool>. It can be connected to an sc_core::sc_port<bool> or a TargetSignalSocket<bool>. Modifications to the interal QEMU GPIO will be propagated through the socket.

If this socket happens to be connected to a QemuTargetSignalSocket, the propagation is done directly within QEMU and do not go through the SystemC kernel. Note that this is only true if the GPIOs wrapped by both this socket and the remote socket lie in the same QEMU instance.

4.18.2 Member Function Documentation

4.18.2.1 init()

```
void QemuInitiatorSignalSocket::init (
            qemu::Device dev,
            int gpio_idx )  [inline]
```

Initialize this socket with a device and a GPIO index.

This method initializes the socket using the given QEMU device and the corresponding GPIO index in this device. See the QEMU API and the device you want to wrap to know what index to use here.

Parameters

in	*dev*	The QEMU device
in	*gpio_idx*	The GPIO index within the device

4.18.2.2 init_named()

```
void QemuInitiatorSignalSocket::init_named (
```

```
            qemu::Device dev,
            const char * gpio_name,
            int gpio_idx )  [inline]
```

Initialize this socket with a device, a GPIO namespace, and a GPIO index.

This method initializes the socket using the given QEMU device and the corresponding GPIO (namespace, index) pair in this device. See the QEMU API and the device you want to wrap to know what namespace/index to use here.

Parameters

in	*dev*	The QEMU device
in	*gpio_name*	The GPIO namespace within the device
in	*gpio_idx*	The GPIO index within the device

4.18.2.3 init_sbd()

```
void QemuInitiatorSignalSocket::init_sbd (
            qemu::SysBusDevice sbd,
            int gpio_idx )  [inline]
```

Initialize this socket with a QEMU SysBusDevice, and a GPIO index.

This method initializes the socket using the given QEMU SysBusDevice (SBD) and the corresponding GPIO index) in this SBD. See the QEMU API and the SBD you want to wrap to know what index to use here.

Parameters

in	*sbd*	The QEMU SysBusDevice
in	*gpio_idx*	The GPIO index within the SBD

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

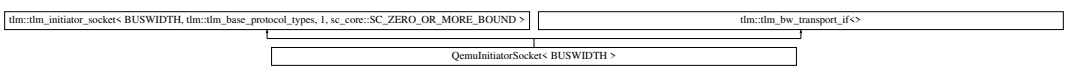
- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/ports/initiator-signal-socket.h

4.19 QemuInitiatorSocket< BUSWIDTH > Class Template Reference

TLM-2.0 initiator socket specialisation for QEMU AddressSpace mapping.

```
#include <initiator.h>
```

Inheritance diagram for QemuInitiatorSocket< BUSWIDTH >:



Public Types

- using **TlmInitiatorSocket** = tlm::tlm_initiator_socket< BUSWIDTH, tlm::tlm_base_protocol_types, 1, sc_core::SC_ZERO_OR_MORE_BOUND >
- using **TlmPayload** = tlm::tlm_generic_payload
- using **MemTxResult** = qemu::MemoryRegionOps::MemTxResult
- using **MemTxAttrs** = qemu::MemoryRegionOps::MemTxAttrs
- using **DmiRegion** = QemuInstanceDmiManager::DmiRegion
- using **DmiRegionAlias** = QemuInstanceDmiManager::DmiRegionAlias
- using **DmiRegionAliasKey** = uint64_t

Public Member Functions

- **QemuInitiatorSocket** (const char *name, QemuInitiatorIface &initiator, QemuInstance &inst)
- void **init** (qemu::Device &dev, const char *prop)
- void **cancel_all** ()
- virtual tlm::tlm_sync_enum **nb_transport_bw** (tlm::tlm_generic_payload &trans, tlm::tlm_phase &phase, sc_core::sc_time &t)
- virtual void **invalidate_direct_mem_ptr** (sc_dt::uint64 start_range, sc_dt::uint64 end_range)

Protected Member Functions

- void **init_payload** (TlmPayload &trans, tlm::tlm_command command, uint64_t addr, uint64_t *val, unsigned int size)
- DmiRegionAliasKey **get_dmi_region_alias_key** (const tlm::tlm_dmi &info)
- DmiRegionAliasKey **get_dmi_region_alias_key** (const DmiRegionAlias &alias)
- void **add_dmi_mr_alias** (DmiRegionAlias &alias)
- void **del_dmi_mr_alias** (const DmiRegionAlias &alias)
- DmiRegionAlias * **request_dmi_region** (TlmPayload &trans)
- void **check_dmi_hint** (TlmPayload &trans)
- void **check_qemu_mr_hint** (TlmPayload &trans)
- void **do_regular_access** (TlmPayload &trans)
- void **do_debug_access** (TlmPayload &trans)
- MemTxResult **qemu_io_access** (tlm::tlm_command command, uint64_t addr, uint64_t *val, unsigned int size, MemTxAttrs attrs)
- MemTxResult **qemu_io_read** (uint64_t addr, uint64_t *val, unsigned int size, MemTxAttrs attrs)
- MemTxResult **qemu_io_write** (uint64_t addr, uint64_t val, unsigned int size, MemTxAttrs attrs)

Protected Attributes

- QemuInstance & **m_inst**
- QemuInitiatorIface & **m_initiator**
- qemu::Device **m_dev**
- gs::RunOnSysC **m_on_sysc**
- qemu::MemoryRegion **m_root**
- std::map< DmiRegionAliasKey, DmiRegionAlias > **m_dmi_aliases**

4.19.1 Detailed Description

template<unsigned int BUSWIDTH = 32>
class QemuInitiatorSocket< BUSWIDTH >

TLM-2.0 initiator socket specialisation for QEMU AddressSpace mapping.

This class is used to expose a QEMU AddressSpace object as a standard TLM-2.0 initiator socket. It creates a root memory region to map the whole address space, receives I/O accesses to it and forwards them as standard TLM-2.0 transactions.

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/ports/initiator.h

4.20 QemuInstance Class Reference

This class encapsulates a libqemu-cxx qemu::LibQemu instance. It handles QEMU parameters and instance initialization.

`#include <qemu-instance.h>`

Public Types

- enum **TcgMode** { **TCG_UNSPECIFIED**, **TCG_SINGLE**, **TCG_SINGLE_COROUTINE**, **TCG_MULTI** }
- enum **IcountMode** { **ICOUNT_UNSPECIFIED**, **ICOUNT_OFF**, **ICOUNT_ON** }
- using **Target** = qemu::Target
- using **LibLoader** = qemu::LibraryLoaderIface

Public Member Functions

- **QemuInstance** (LibLoader &loader, Target t)
- **QemuInstance** (const QemuInstance &)=delete
- **QemuInstance** (QemuInstance &&)=delete
- bool **operator==** (const QemuInstance &b) const
- bool **operator!=** (const QemuInstance &b) const
- void add_arg (const char *arg)

  Add a command line argument to the qemu instance.
- void set_tcg_mode (TcgMode m)

  Set the desired TCG mode for this instance.
- void set_icount_mode (IcountMode m, int mips_shift)

  Set the desired icount mode for this instance.
- void init ()

  Initialize the QEMU instance.
- bool is_inited () const

  Returns true if the instance is initialized.
- qemu::LibQemu & get ()

  Returns the underlying qemu::LibQemu instance.
- LockedQemuInstanceDmiManager get_dmi_manager ()

  Returns the locked QemuInstanceDmiManager instance.

Protected Member Functions

- void **push_default_args** ()
- void **push_icount_mode_args** ()
- void **push_tcg_mode_args** ()

Protected Attributes

- qemu::LibQemu **m_inst**
- QemuInstanceDmiManager **m_dmi_mgr**
- TcgMode **m_tcg_mode** = TCG_UNSPECIFIED
- IcountMode **m_icount_mode** = ICOUNT_UNSPECIFIED
- int **m_icount_mips** = 0

4.20.1 Detailed Description

This class encapsulates a libqemu-cxx qemu::LibQemu instance. It handles QEMU parameters and instance initialization.

4.20.2 Member Function Documentation

4.20.2.1 add_arg()

```
void QemuInstance::add_arg (
            const char * arg )  [inline]
```

Add a command line argument to the qemu instance.

This method may only be called before the instance is initialized.

4.20.2.2 get()

```
qemu::LibQemu& QemuInstance::get ( )  [inline]
```

Returns the underlying qemu::LibQemu instance.

Returns the underlying qemu::LibQemu instance. If the instance hasn't been initialized, init is called just before returning the instance.

4.20.2.3 get_dmi_manager()

```
LockedQemuInstanceDmiManager QemuInstance::get_dmi_manager ( )  [inline]
```

Returns the locked QemuInstanceDmiManager instance.

Note: we rely on RVO here so no copy happen on return (this is enforced by the fact that the LockedQemu↩ InstanceDmiManager copy constructor is deleted).

4.20.2.4 init()

```
void QemuInstance::init ( )  [inline]
```

Initialize the QEMU instance.

Initialize the QEMU instance with the set TCG and icount mode. If the TCG mode hasn't been set, it defaults to TCG_SINGLE. If icount mode hasn't been set, it defaults to ICOUNT_OFF.

The instance should not already be initialized when calling this method.

4.20.2.5 set_icount_mode()

```
void QemuInstance::set_icount_mode (
            IcountMode m,
            int mips_shift )  [inline]
```

Set the desired icount mode for this instance.

This method is called by CPU instances to specify the desired icount mode according to the synchronization policy in use. All CPUs should use the same mode.

This method should be called before the instance is initialized.

Parameters

in	*m*	The desired icount mode
in	*mips_shift*	The QEMU icount shift parameter. It sets the virtual time an instruction takes to execute to 2^(mips_shift) ns.

4.20.2.6 set_tcg_mode()

```
void QemuInstance::set_tcg_mode (
            TcgMode m )  [inline]
```

Set the desired TCG mode for this instance.

This method is called by CPU instances to specify the desired TCG mode according to the synchronization policy in use. All CPUs should use the same mode (meaning they should all use synchronization policies compatible one with the other).

This method should be called before the instance is initialized.

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/qemu-instance.h

4.21 QemuInstanceDmiManager Class Reference

Handles the DMI regions at the QEMU instance level.

`#include <dmi-manager.h>`

Classes

- class DmiRegion

  a DMI region
- class DmiRegionAlias

  An alias to a DMI region.
- class QemuContainer

Public Member Functions

- **QemuInstanceDmiManager** (qemu::LibQemu &inst)
- **QemuInstanceDmiManager** (const QemuInstanceDmiManager &)=delete
- **QemuInstanceDmiManager** (QemuInstanceDmiManager &&a)
- DmiRegionAlias get_new_region_alias (const tlm::tlm_dmi &info)

  Create a new alias for the DMI region designated by `info`

Protected Member Functions

- DmiRegion::Ptr **create_region** (const tlm::tlm_dmi &info)
- DmiRegion::Ptr **get_region** (const tlm::tlm_dmi &info)

Protected Attributes

- qemu::LibQemu & **m_inst**
- std::mutex **m_mutex**
- std::map< DmiRegion::Key, std::weak_ptr< DmiRegion > > **m_regions**

Friends

- class **LockedQemuInstanceDmiManager**

4.21.1 Detailed Description

Handles the DMI regions at the QEMU instance level.

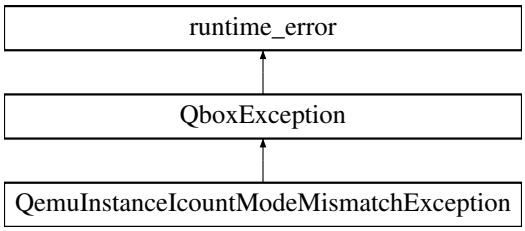
This class handles the DMI regions at the level of a QEMU instance. For a given DMI region, we need to use a unique memory region (called the global memory region, in a sense that it is global to all the CPUs in the instance). Each CPU is then supposed to create an alias to this region to be able to access it. This is required to ensure QEMU sees this region as a unique piece of memory. Creating multiple regions mapping to the same host address leads QEMU into thinking that those are different data, and it won't properly invalidate corresponding TBs if CPUs do SMC (self modifying code).

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/dmi-manager.h

4.22 QemuInstanceIcountModeMismatchException Class Reference

Inheritance diagram for QemuInstanceIcountModeMismatchException:



Additional Inherited Members

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/qemu-instance.h

4.23 QemuInstanceManager Class Reference

QEMU instance manager class.

`#include <qemu-instance.h>`

Public Types

- using **Target** = qemu::Target
- using **LibLoader** = qemu::LibraryLoaderIface

Public Member Functions

- QemuInstanceManager ()

  Construct a QemuInstanceManager. The manager will use the default library loader provided by libqemu-cxx.
- QemuInstanceManager (LibLoader ∗loader)

  Construct a QemuInstanceManager by providing a custom library loader.
- QemuInstance & new_instance (Target t)

  Returns a new QEMU instance for target t.

Protected Attributes

- LibLoader ∗ **m_loader**
- std::vector< std::reference_wrapper< QemuInstance > > **m_insts**

4.23.1 Detailed Description

QEMU instance manager class.

This class manages QEMU instances. It allows to create instances using the same library loader, thus allowing multiple instances of the same library being loaded.

4.23.2 Constructor & Destructor Documentation

4.23.2.1 QemuInstanceManager()

```
QemuInstanceManager::QemuInstanceManager (
            LibLoader * loader )  [inline]
```

Construct a QemuInstanceManager by providing a custom library loader.

Parameters

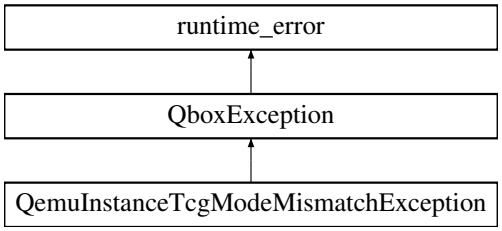
in	*loader*	The custom loader

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/qemu-instance.h

4.24 QemuInstanceTcgModeMismatchException Class Reference

Inheritance diagram for QemuInstanceTcgModeMismatchException:



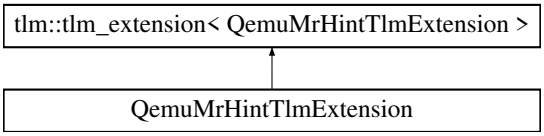
Additional Inherited Members

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/qemu-instance.h

4.25 QemuMrHintTlmExtension Class Reference

Inheritance diagram for QemuMrHintTlmExtension:



Public Member Functions

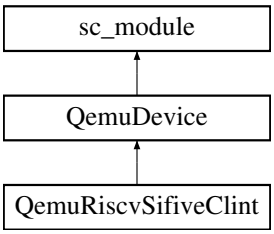
- **QemuMrHintTlmExtension** (const QemuMrHintTlmExtension &)=default
- **QemuMrHintTlmExtension** (qemu::MemoryRegion mr, uint64_t offset)
- virtual tlm_extension_base * **clone** () const override
- virtual void **copy_from** (tlm_extension_base const &ext) override
- qemu::MemoryRegion **get_mr** () const
- uint64_t **get_offset** () const

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/tlm-extensions/qemu-mr-hint.h

4.26 QemuRiscvSifiveClint Class Reference

Inheritance diagram for QemuRiscvSifiveClint:



Public Member Functions

- **QemuRiscvSifiveClint** (sc_core::sc_module_name nm, QemuInstance &inst)
- void **before_end_of_elaboration** () override
- void **end_of_elaboration** () override

Public Attributes

- cci::cci_param< unsigned int > **p_num_harts**
- cci::cci_param< uint64_t > **p_sip_base**
- cci::cci_param< uint64_t > **p_timecmp_base**
- cci::cci_param< uint64_t > **p_time_base**
- cci::cci_param< bool > **p_provide_rdtime**
- cci::cci_param< uint64_t > **p_aperture_size**
- QemuTargetSocket **socket**

Protected Attributes

- uint64_t **m_aperture_size**
- int **m_num_harts**

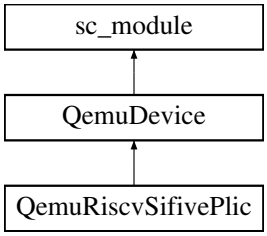
Additional Inherited Members

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/components/irq-ctrl/clint-sifive.h

4.27 QemuRiscvSifivePlic Class Reference

Inheritance diagram for QemuRiscvSifivePlic:



Public Member Functions

- **QemuRiscvSifivePlic** (sc_core::sc_module_name nm, QemuInstance &inst)
- void **before_end_of_elaboration** () override
- void **end_of_elaboration** () override

Public Attributes

- cci::cci_param< unsigned int > **p_num_sources**
- cci::cci_param< unsigned int > **p_num_priorities**
- cci::cci_param< uint64_t > **p_priority_base**
- cci::cci_param< uint64_t > **p_pending_base**
- cci::cci_param< uint64_t > **p_enable_base**
- cci::cci_param< uint64_t > **p_enable_stride**
- cci::cci_param< uint64_t > **p_context_base**
- cci::cci_param< uint64_t > **p_context_stride**
- cci::cci_param< uint64_t > **p_aperture_size**
- cci::cci_param< std::string > **p_hart_config**
- QemuTargetSocket **socket**
- sc_core::sc_vector< QemuTargetSignalSocket > **irq_in**

Additional Inherited Members

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

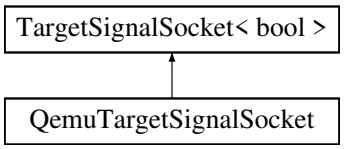
- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/components/irq-ctrl/plic-sifive.h

4.28 QemuTargetSignalSocket Class Reference

A QEMU input GPIO exposed as a TargetSignalSocket<bool>

`#include <target-signal-socket.h>`

Inheritance diagram for QemuTargetSignalSocket:



Public Member Functions

- **QemuTargetSignalSocket** (const char *name)
- void init (qemu::Device dev, int gpio_idx)

  Initialize this socket with a device and a GPIO index.
- void init_named (qemu::Device dev, const char *gpio_name, int gpio_idx)

  Initialize this socket with a device, a GPIO namespace, and a GPIO index.
- qemu::Gpio get_gpio ()

  Returns the GPIO wrapped by this socket.
- void notify ()

  Force a notification on the default event.

Protected Member Functions

- void **value_changed_cb** (const bool &val)
- void **init_with_gpio** (qemu::Gpio gpio)

Protected Attributes

- qemu::Gpio **m_gpio_in**

4.28.1 Detailed Description

A QEMU input GPIO exposed as a TargetSignalSocket<bool>

This class exposes an input GPIO of a QEMU device as a TargetSignalSocket<bool>. It can be connected to an sc_core::sc_port<bool> or a TargetInitiatorSocket<bool>. Modifications to this socket will be reported to the wrapped GPIO.

4.28.2 Member Function Documentation

4.28.2.1 get_gpio()

```
qemu::Gpio QemuTargetSignalSocket::get_gpio ( )  [inline]
```

Returns the GPIO wrapped by this socket.

Returns

the GPIO wrapped by this socket

4.28.2.2 init()

```
void QemuTargetSignalSocket::init (
            qemu::Device dev,
            int gpio_idx )  [inline]
```

Initialize this socket with a device and a GPIO index.

This method initializes the socket using the given QEMU device and the corresponding GPIO index in this device. See the QEMU API and the device you want to wrap to know what index to use here.

Parameters

in	*dev*	The QEMU device
in	*gpio_idx*	The GPIO index within the device

4.28.2.3 init_named()

```
void QemuTargetSignalSocket::init_named (
            qemu::Device dev,
            const char * gpio_name,
            int gpio_idx )  [inline]
```

Initialize this socket with a device, a GPIO namespace, and a GPIO index.

This method initializes the socket using the given QEMU device and the corresponding GPIO (namespace, index) pair in this device. See the QEMU API and the device you want to wrap to know what namespace/index to use here.

Parameters

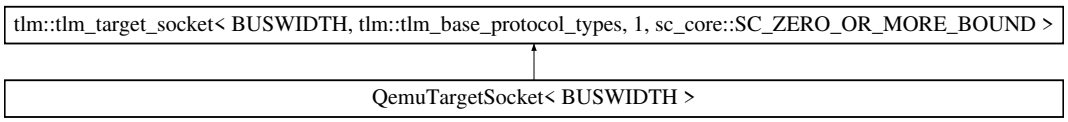
in	*dev*	The QEMU device
in	*gpio_name*	The GPIO namespace within the device
in	*gpio_idx*	The GPIO index within the device

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/ports/target-signal-socket.h

4.29 QemuTargetSocket< BUSWIDTH > Class Template Reference

Inheritance diagram for QemuTargetSocket< BUSWIDTH >:



Public Types

- using **TlmTargetSocket** = tlm::tlm_target_socket< BUSWIDTH, tlm::tlm_base_protocol_types, 1, sc_core↩ ::SC_ZERO_OR_MORE_BOUND >
- using **TlmPayload** = tlm::tlm_generic_payload

Public Member Functions

- **QemuTargetSocket** (const char *name, QemuInstance &inst)
- void **init** (qemu::SysBusDevice sbd, int mmio_idx)
- void **init_with_mr** (qemu::MemoryRegion mr)

Protected Attributes

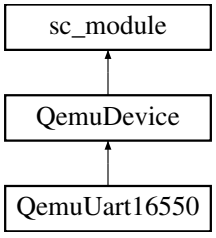
- TlmTargetToQemuBridge **m_bridge**
- QemuInstance & **m_inst**
- qemu::SysBusDevice **m_sbd**

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/ports/target.h

4.30 QemuUart16550 Class Reference

Inheritance diagram for QemuUart16550:



Public Member Functions

- **QemuUart16550** (const sc_core::sc_module_name &n, QemuInstance &inst)
- void **before_end_of_elaboration** () override
- void **end_of_elaboration** () override

Public Attributes

- QemuTargetSocket **socket**
- QemuInitiatorSignalSocket **irq_out**

Protected Attributes

- qemu::Chardev **m_chardev**
- cci::cci_param< unsigned int > **p_baudbase**
- cci::cci_param< unsigned int > **p_regshift**

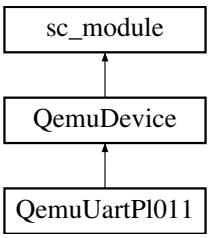
Additional Inherited Members

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/components/uart/16550.h

4.31 QemuUartPl011 Class Reference

Inheritance diagram for QemuUartPl011:



Public Member Functions

- **QemuUartPl011** (const sc_core::sc_module_name &n, QemuInstance &inst)
- void **before_end_of_elaboration** () override
- void **end_of_elaboration** () override

Public Attributes

- QemuTargetSocket **socket**
- QemuInitiatorSignalSocket **irq_out**

Protected Attributes

- qemu::Chardev **m_chardev**
- gs::async_event **m_ext_ev**

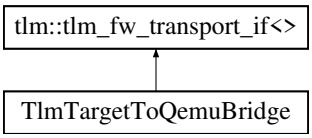
Additional Inherited Members

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/components/uart/pl011.h

4.32 TlmTargetToQemuBridge Class Reference

Inheritance diagram for TlmTargetToQemuBridge:



Public Types

- using **MemTxAttrs** = qemu::MemoryRegion::MemTxAttrs
- using **MemTxResult** = qemu::MemoryRegion::MemTxResult
- using **TlmPayload** = tlm::tlm_generic_payload

Public Member Functions

- void **init** (qemu::SysBusDevice sbd, int mmio_idx)
- void **init_with_mr** (qemu::MemoryRegion mr)
- virtual void **b_transport** (TlmPayload &trans, sc_core::sc_time &t)
- virtual tlm::tlm_sync_enum **nb_transport_fw** (TlmPayload &trans, tlm::tlm_phase &phase, sc_core::sc_time &t)
- virtual bool **get_direct_mem_ptr** (TlmPayload &trans, tlm::tlm_dmi &dmi_data)
- virtual unsigned int **transport_dbg** (TlmPayload &trans)

Protected Member Functions

- void **init_as** ()
- qemu::Cpu **push_current_cpu** (TlmPayload &trans)
- void **pop_current_cpu** (qemu::Cpu cpu)

Protected Attributes

- qemu::MemoryRegion **m_mr**
- std::shared_ptr< qemu::AddressSpace > **m_as**

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/ports/target.h

Index