gssync

Generated by Doxygen 1.9.1

Chapter 1

Main Page

1.0.1 GreenSocs Synchronization Library

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.

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

1.1 GreenSocs Build and make system

1.2 How to build

This project may be built using cmake

```
cmake -B BUILD;cd BUILD; make -j
```

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
```

1.2.1 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>_SOURCE_DIR`. Hence you may wish to use your own copy of SystemC

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.

1.2.1.1 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.

1.2.2 More documentation

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

1.2.3 GreenSocs Basic SystemC utility library

This is the GreenSocs basic utilities library. It contains utility functions for CCI and simple logging functions.

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.

1.2.3.1 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.

1.2.3.2 Using yaml

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
```

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

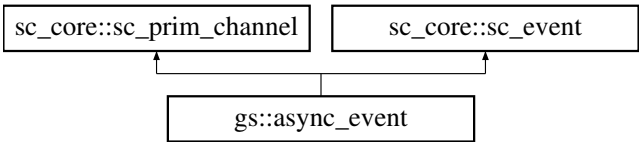
Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

Class Documentation

4.1 gs::async_event Class Reference

Inheritance diagram for gs::async_event:



Public Member Functions

- **async_event** (bool start_attached=true)
- void **async_notify** ()
- void **notify** (sc_core::sc_time delay=sc_core::sc_time(sc_core::SC_ZERO_TIME))
- void **async_attach_suspending** ()
- void **async_detach_suspending** ()
- void **enable_attach_suspending** (bool e)

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

- /Users/mark/work/libgs/tmp/libgssync/include/greensocs/libgssync/async_event.h

4.2 gs::global_pause Class Reference

Public Member Functions

- **global_pause** (const global_pause &)=delete
- void **suspendable** ()
- void **unsuspendable** ()
- void **unsuspend_all** ()
- void **suspend_all** ()
- bool **attach_suspending** (sc_core::sc_prim_channel ∗p)
- bool **detach_suspending** (sc_core::sc_prim_channel ∗p)
- void **async_wakeup** ()

Static Public Member Functions

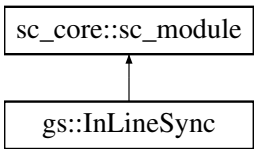
- static global_pause & **get** ()

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

- /Users/mark/work/libgs/tmp/libgssync/include/greensocs/libgssync/pre_suspending_sc_support.h
- /Users/mark/work/libgs/tmp/libgssync/src/pre_suspending_sc_support.cc

4.3 gs::InLineSync Class Reference

Inheritance diagram for gs::InLineSync:



Public Member Functions

- **SC_HAS_PROCESS** (InLineSync)
- **InLineSync** (const sc_core::sc_module_name &name)
- void **b_transport** (tlm::tlm_generic_payload &trans, sc_core::sc_time &delay)
- void **get_direct_mem_ptr** (tlm::tlm_generic_payload &trans, tlm::tlm_dmi &dmi_data)
- void **transport_dgb** (tlm::tlm_generic_payload &trans)
- void **invalidate_direct_mem_ptr** (sc_dt::uint64 start_range, sc_dt::uint64 end_range)

Public Attributes

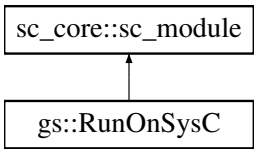
- tlm_utils::simple_target_socket< InLineSync > **target_socket**
- tlm_utils::simple_initiator_socket< InLineSync > **initiator_socket**

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

- /Users/mark/work/libgs/tmp/libgssync/include/greensocs/libgssync/inlinesync.h

4.4 gs::RunOnSysC Class Reference

Inheritance diagram for gs::RunOnSysC:



Public Member Functions

- **RunOnSysC** (const sc_core::sc_module_name &n=sc_core::sc_module_name("run-on-sysc"))
- void **end_of_simulation** ()
- void **fork_on_systemc** (std::function< void()> job_entry)
- void **run_on_sysc** (std::function< void()> job_entry, bool complete=true)

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

- /Users/mark/work/libgs/tmp/libgssync/include/greensocs/libgssync/runonsysc.h

4.5 gs::semaphore Class Reference

Public Member Functions

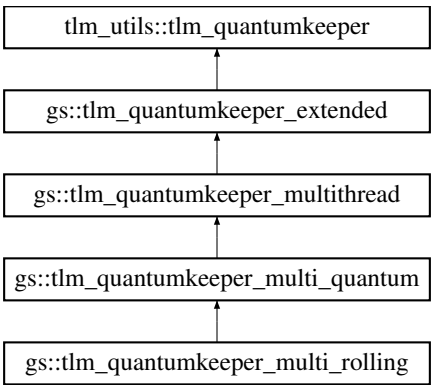
- void **notify** ()
- void **wait** ()
- bool **try_wait** ()

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

- /Users/mark/work/libgs/tmp/libgssync/include/greensocs/libgssync/semaphore.h

4.6 gs::tlm_quantumkeeper_extended Class Reference

Inheritance diagram for gs::tlm_quantumkeeper_extended:



Public Member Functions

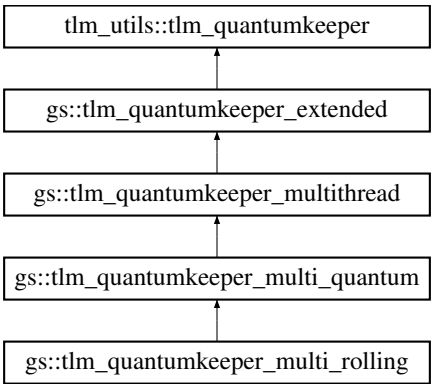
- virtual sc_core::sc_time **time_to_sync** ()
- virtual void **stop** ()
- virtual void **start** (std::function< void()> job=nullptr)
- virtual SyncPolicy::Type **get_thread_type** () const
- virtual bool **need_sync** ()
- virtual bool **need_sync** () const override
- virtual void **run_on_systemc** (std::function< void()> job)

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

- /Users/mark/work/libgs/tmp/libgssync/include/greensocs/libgssync/qk_extendedif.h

4.7 gs::tlm_quantumkeeper_multi_quantum Class Reference

Inheritance diagram for gs::tlm_quantumkeeper_multi_quantum:



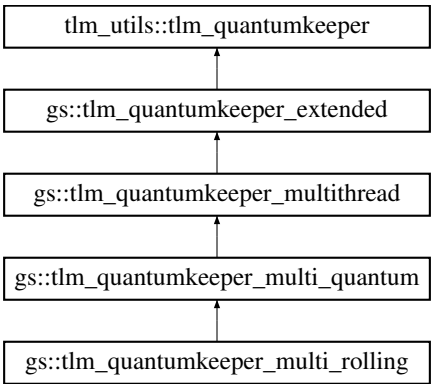
Additional Inherited Members

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

- /Users/mark/work/libgs/tmp/libgssync/include/greensocs/libgssync/qkmulti-quantum.h

4.8 gs::tlm_quantumkeeper_multi_rolling Class Reference

Inheritance diagram for gs::tlm_quantumkeeper_multi_rolling:



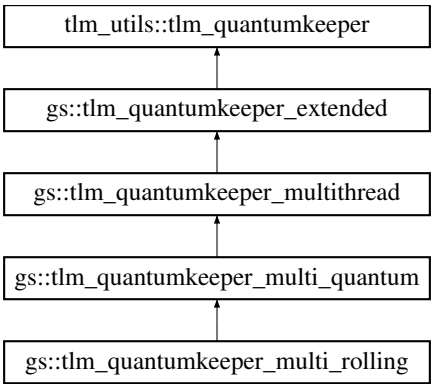
Additional Inherited Members

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

- /Users/mark/work/libgs/tmp/libgssync/include/greensocs/libgssync/qkmulti-rolling.h

4.9 gs::tlm_quantumkeeper_multithread Class Reference

Inheritance diagram for gs::tlm_quantumkeeper_multithread:



Public Member Functions

- virtual SyncPolicy::Type **get_thread_type** () const override
- virtual void **start** (std::function< void()> job=nullptr) override
- virtual void **stop** () override
- virtual sc_core::sc_time **time_to_sync** () override
- void **inc** (const sc_core::sc_time &t) override
- void **set** (const sc_core::sc_time &t) override
- virtual void **sync** () override
- void **reset** () override
- sc_core::sc_time **get_current_time** () const override
- sc_core::sc_time **get_local_time** () const override
- virtual bool **need_sync** () override

Protected Types

- enum **jobstates** { **NONE** , **RUNNING** , **STOPPED** }

Protected Member Functions

- virtual bool **is_sysc_thread** () const

Protected Attributes

- enum gs::tlm_quantumkeeper_multithread::jobstates **status**
- async_event **m_tick**

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

- /Users/mark/work/libgs/tmp/libgssync/include/greensocs/libgssync/qkmultithread.h
- /Users/mark/work/libgs/tmp/libgssync/src/qkmultithread.cc

Index